Aalto University
School of Electrical Engineering
Department of Communications and Networking

Jesús Llorente Santos

Private Realm Gateway

Thesis submitted in partial fulfillment of the requirements for the degree of Master of Science in Technology.

Espoo 08.11.2012

Thesis supervisor: Prof. Raimo Kantola
Aalto University

Thesis instructor: D. Sc. (Tech.) Nicklas Beijar
Aalto University

Aalto University School of Science and Technology **Abstract of Master's Thesis**

Author: Jesús Llorente Santos

Name of the Thesis: Private Realm Gateway

Date: 08.11.2012 **Language:** English **Number of pages:** VI + 115

School: School of Electrical Engineering

Department: Department of Communications and Networking

Professorship: Networking Technology **Code:** S.38

Supervisor: Prof. Raimo Kantola, Aalto University

Instructor: D. Sc. (Tech.) Nicklas Beijar, Aalto University

The IPv4 address exhaustion has been a global concern for the last two decades. The increased number of connected users and services has depleted almost entirely the addresses available. There have been several attempts to solve this problem. Chronologically they are Classless Inter-Domain Routing (CIDR), Network Address Translation (NAT) and a new version of the IP protocol, IPv6.

The adoption of NAT introduced the separation of private and public realms. NAT devices allow the hosts located in the private realm to connect with hosts or services in the public realm by sharing a public IP address. NAT also provides the foremost kind of firewall blocking incoming connections towards the private realms and introducing the reachability problem. Although several alternatives have been developed to overcome this issue, none of them are exempt of drawbacks.

This thesis introduces a new concept that solves the reachability problem introduced by NAT. The solution is called Private Realm Gateway (PRGW). The main component is called Circular Pool and it uses a limited number of public IP addresses to enable end-to-end communication to most applications. Other applications require the use of Application Layer Gateway (ALG) or proxy servers to grant connectivity.

The evaluation of the prototype proves the concept and the implementation highly successful. The Private Realm Gateway provides mechanisms to overcome the reachability problem and also contributes to the solution of the address exhaustion problem.

Keywords: IP, NAT, Traversal, Reachability, PRGW, CES

Aalto-yliopiston teknillinen korkeakoulu **Diplomityön tiivistelmä**

Tekijä: Jesús Llorente Santos

Työn nimi: Yksityisen alueen yhdyskäytävä

Päivämäärä: 08.11.2012 **Kieli:** englanti **Sivumäärä:** VI + 115

Korkeakoulu: Sähkötekniikan korkeakoulu

Laitos: Tietoliikenne- ja tietoverkkotekniikan laitos

Professuuri: Tietoverkkotekniikka **Koodi:** S.38

Valvoja: Prof. Raimo Kantola, Aalto-yliopisto

Ohjaaja: TkT Nicklas Beijar, Aalto-yliopisto

IPv4-osoitteiden loppuminen on ollut maailmanlaajuinen huoli jo viimeisen kahden vuosikymmenen ajan. Lisääntynyt käyttäjien ja palvelujen lukumäärä on kuluttanut jo lähes kaikki mahdolliset osoitteet. Useita ratkaisuja on esitetty ongelman ratkaisemiseksi. Aikajärjestyksessä nämä ovat luokaton reititys (CIDR), osoitteenmuunnos (NAT) ja uusi versio IP protokollasta, IPv6.

Osoitteenmuunnoksen käyttöönottaminen jakoi alueet yksityisiin ja julkisiin. NAT laitteet sallivat yksityisen verkon käyttäjien kommunikoida julkisen verkon käyttäjien kanssa jaetun IP osoitteen välityksellä. NAT toimii myös yksinkertaisena palomuurina estäen sisääntulevan liikenteen ja siten aiheuttaen ongelmia saavutettavuuden kanssa. Useista ratkaisuista huolimatta, yksikään ratkaisu ei ole täysin ongelmaton.

Tässä työssä esitellään ratkaisu osoitteenmuutoksen aiheuttamaan saavutettavuusongelmaan. Ratkaisu on nimeltään Yksityisen Alueen Yhdyskäytävä (PRGW). Ratkaisun pääkomponentti on nimeltään kiertävä (renkaanmuotoinen) osoitevaranto joka käyttää rajoitettua määrää julkisia osoitteita mahdollistaen päästä-päähän kommunikoinnin useimmille sovelluksille. Loput sovellukset tarvitsevat sovellustason yhdyskäytävän tai välipalvelimen liitettävyyden luomiseksi.

Prototyypin arviointi todistaa teorian ja toteutuksen toimivan erittäin hyvin. Yksityisen alueen yhdyskäytävä tarjoaa mekanismit saavutettavuuden ratkaisemiseksi ja samalla edistää ratkaisua osoitteiden loppumiseen.

Avainsanat: IP, NAT, Traversal, saavutettavuus, PRGW, CES

Acknowledgements

This Master's Thesis has been done for the Department of Communications and Networking, School of Electrical Engineering of Aalto University in April 2011 – October 2012. This thesis is part of a larger project regarding Customer Edge Switching integrated within the MEVICO project.

I want to thank my supervisor, Professor Raimo Kantola, for the opportunity to work in this project. His profound knowledge and valuable insight was crucial for the realization of this thesis.

I would also like to thank my instructor, Nicklas Beijar, for listening to my *concerns* and all the discussions during the past year. Nicklas provided me with great ideas and incalculable help during the writing process.

My gratitude also goes to Petri Leppäaho, with whom I forged a magnificent friendship while working on the same project.

I want to thank my friends, they know who they are. Their support and valuable insight was very important to me; the RFCs 1149 and 2795 helped me along the way.

I would like to thank my dear *Vaidute*, for believing in me during all these years and most important, for being.

Finally, I want to thank my family for helping me during all these years abroad and receiving me with open arms whenever I would be back at home. Despite their recurrent IT problems my love is with them.

8 November 2012, Espoo, Finland.

Jesús Llorente Santos

Table of Contents

List of Figures

List of Tables

Abbreviations

ALG	Application Layer Gateway
CES	Customer Edge Switching
CETP	Customer Edge Traversal Protocol
CIDR	Classless Inter-Domain Routing
DDNS	Dynamic DNS
DHCP	Dynamic Host Configuration Protocol
DMZ	Demilitarized Zone
DNS	Domain Name System
DoD	Department of Defense
DoS/DDoS	Denial of Service / Distributed Denial of Service
FQDN	Fully Qualified Domain Name
FTP	File Transfer Protocol
HTTP	Hypertext Transfer Protocol
IANA	Internet Assigned Numbers Authority
ICE	Interactive Connectivity Establishment
ICMP	Internet Control Message Protocol
ICT	Information and Communications Technologies
IETF	Internet Engineering Task Force
IP	Internet Protocol
IPsec	Internet Protocol Security
ISO	International Organization for Standardization
ISP	Internet Service Provider
ITU-T	International Telecommunication Union-Telecommunication
NAT	Network Address Translation
NTP	Network Time Protocol
OSI	Open Systems Interconnection
PDU	Protocol Data Unit
PRGW	Private Realm Gateway
RFC	Request for Comments
SDP	Session Description Protocol
SIP	Session Initiation Protocol
SPN	Service Provider Networks
SSH	Secure Shell
SSL	Secure Sockets Layer
STUN	Session Traversal Utilities for NAT
TCP	Transmission Control Protocol
TFTP	Trivial File Transfer Protocol
TLS	Transport Layer Security
TTL	Time to Live
TURN	Traversal Using Relays around NAT
UDP	User Datagram Protocol
UN	User Networks
UPnP	Universal Plug and Play
URL	Uniform Resource Locators
UTC	Coordinated Universal Time
VoIP	Voice over IP
WWW	World Wide Web

1. Introduction

During the decade of the 90s, the Internet expanded dramatically changing from a scientific and governmental research network to a commercial and consumer marketplace. At the same time, traditional services such as media initiated a deep process of change towards electronic platforms. New services were launched and companies emerged overnight attracted for the huge market opportunities. Internet entailed a worldwide revolution achieving a great share in household penetration. From that moment on, the term communication changed forever, bringing the opportunity of interacting with anyone, anywhere, anytime.

During the past 20 years there have been many significant technological breakthroughs encouraged by Internet Service Providers (ISP), mobile operators and manufacturers. The transformation and the innovation of services have affected the media and the traditional communication model. Especially multimedia broadcasting and file sharing services are greatly responsible for increasing the data traffic as well as the available content. As a result, these new applications offer and consume contents and services independently. Figure 1.1 represents the evolution over the past ten years in Internet users and telephony subscriptions.

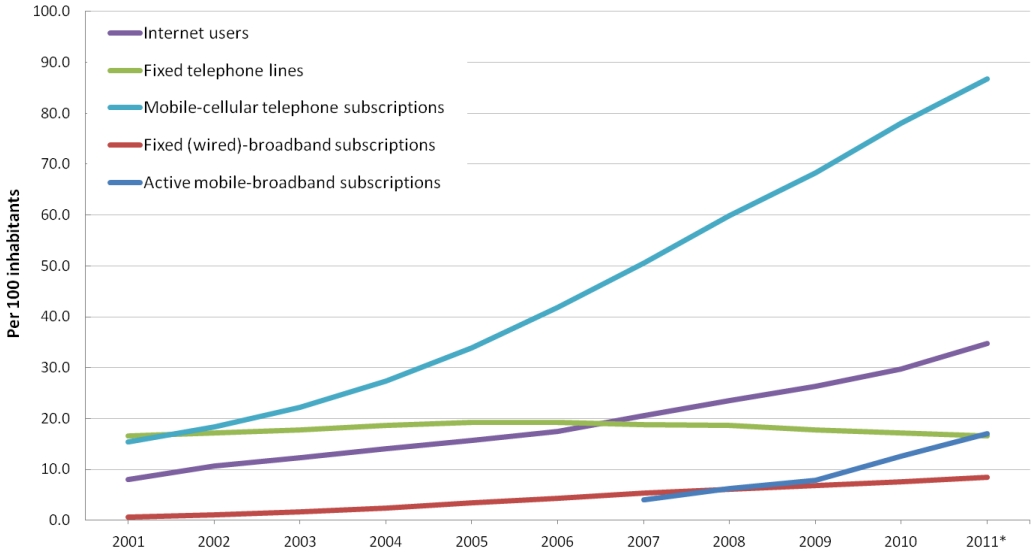


FIGURE 1.1 ITU-T GLOBAL ICT DEVELOPMENTS 2001-2011

In 2006 a remarkable milestone took place in the ICT sector. For the first time the number of fixed telephone subscribers started to decrease in favor of mobile telephone

subscriptions. The figure reveals how in 2008 the penetration of mobile broadband already outgrows fixed broadband subscription [11]. As of 12/2011, ITU-T statistic reveals that mobile subscription reached 1.2 billion users which represent approximately half of the Internet users. As a consequence, there is a justification for concentrating on the needs of mobile customers while devising new solutions.

1.1 Motivation and Background

Due to its great success, the Internet had to face important transformations in order to survive its own expansion rate. These reasons are the direct effect of the limitations imposed by the Internet Protocol version 4 (IPv4) developed early in the 1980's. These problems are address exhaustion, efficient routing and security. With regard to the first one, the address shortage was and will always be one of the biggest limitations of this protocol. Many solutions were proposed, among them it seems reasonable to bring out Classless Inter-Domain Routing (CIDR) and Network Address Translation (NAT) developed for IPv4 in the short term as well as a new version of the Internet Protocol, version 6 (IPv6), that focuses on the long term.

The adoption of NATs helped to alleviate, but not solve, the address exhaustion problem. A NAT is a device that translates the IP addresses of packets from a private to a public scope and vice versa. NAT enables hosts in private networks to connect to public realms by sharing a public IP address. In addition, it provides the foremost type of firewall protecting the private hosts from public attacks by blocking incoming connections [31]. Consequently, NAT introduces certain restrictions handling these connections known as the *reachability problem*. The question raised is how a host located in a private network can be reached by another in the public realm when there is no explicit mapping in the NAT device for routing these packets.

There have been many proposals throughout the years about how to traverse NAT devices contained in multiple RFCs. These methods are collected under the name of NAT Traversal Protocols and include protocols such as STUN [28], TURN [17] or ICE [29] among others.

Despite all the efforts to solve these problems and due to the late and poor deployment of IPv6, Internet is going through difficult times. As of February 2011, IANA announced the allocation of the last two /8 address pools resulting in the exhaustion of its own pool in favor of APNIC [9] [20]. Due to the rapid demographic and economic expansion of the Asia-Pacific area, it will not take long before all these addresses are allocated.

Raimo Kantola, at the Department of Communications and Networking of Aalto University, conducted a research based on the transition from the end-to-end principle to the trust-to-trust principle [13]. Continuing with that work, Lauri Virtanen presented his M.Sc. Thesis, supervised by Raimo Kantola, implementing a prototype that was called "Customer Edge Switching" [34].

1.2 Research Problem

The purpose of this thesis is to study the separation of private and public realms of addresses and develop a solution that solves the reachability problem introduced by NATs.

Traditionally, a server is required to be located in a public realm to accept incoming connections from a client. The server is then univocally identified by a public IP address that is used by the clients to start a communication. The communication succeeds even if the client is located behind a NAT device. The reachability problem arises when the server is located behind a NAT that does not contain any explicit rules for packet forwarding.

1.3 Objectives and Scope

This thesis continues the research in Customer Edge Switching focusing on the interworking with legacy networks. It introduces a new concept that allows a server located in a private realm, behind a NAT, to receive incoming connections from clients located in public realms or in private realms behind a NAT so that the

NAT provides the public source address for the client. We have called it *Private Realm Gateway (PRGW).*

The proposed solution reuses the existing protocols and operations and does not require changes in either the hosts or the network infrastructure except introducing the PRGW nodes in place of NATs. Interoperability is granted by a series of thorough tests with most common protocols and applications. In addition to provide mechanisms for solving the reachability problem, this thesis also focuses on analyzing the deployment objectives and the scalability of the system.

On the other hand, and despite the topics of security and trust have been considered to be out of the scope, a brief summary of these terms has been included. It is also out of the scope of this thesis to evaluate the impact of nested NAT devices to the network and hosts as well as multihoming and the mobility of users.

1.4 Structure

The thesis is divided into nine chapters.

Chapter 2 covers the current Internet architecture and discusses the IPv4, IPv6, CIDR and DNS concepts. A detailed explanation of NATs is also included. Chapter 3 focuses on the previous work positioning the current research. The concept of Customer Edge Switching is introduced.

Chapter 4 establishes the foundations in terms of design and functional requirements to overcome the problems previously mentioned. Chapter 5 analyzes the differences between the CES and the Internet model, aiming to lay down some valuable insight suitable for the Private Realm Gateway model. Chapter 6 introduces three different models that satisfy to a certain extent the requirements given.

Chapter 7 focuses on the proposed solution called Circular Pool. Chapter 8 evaluates the implemented solution through a set of tests with the most common applications. The chapter also introduces the workarounds developed to overcome connectivity issues. Finally, a performance analysis and summary of the testing are presented. Chapter 9 presents the final conclusions and gives some hints about future research.

2. Internet Protocol Suite

This chapter focuses on explaining the current Internet model. First, some notions of layering and protocols are introduced. Then, the Internet Layer is submitted to a thorough analysis discussing IPv4, IPv6, CIDR and NAT concepts. Afterwards NAT traversal protocols are examined. The chapter finishes with an overview of the name resolution in the Internet.

2.1 Layering and Protocols

The encapsulation of protocols and services is a common practice in order to provide abstraction and layering. Protocols can be defined as standardized sets of operations and procedures for regulating data transmissions between computers or peripherals. Protocols operate under the premise that a layer serves the layer above and is served by the layer below.

The International Organization for Standardization (ISO) is a worldwide institution that promotes proprietary, industrial, and commercial standards similar to DIN, ANSI or ITU. ISO is also responsible for developing the OSI model, taking in different aspects of a communication system and classifying them into seven abstraction layers.

Similarly, the IETF defined the TCP/IP model within the Internet Protocol Suite. According to the RFC 1122 [4], the different functional groups have been categorized into four layers. The protocol layers used in the Internet architecture are the following.

Application Layer
It is located at the top of the architecture, comprising application, presentation and most of the session layer from the OSI model. Some of the protocols included in this layer are FTP, HTTP and DNS.

Transport Layer
It is located under the application layer, offering both connection-oriented and connectionless services. This layer is utterly responsible for the end-to-end data

transfer independently of the underlying network. The layer comprises protocols for real-time (UDP) and non real-time (TCP) traffic along with error control via checksums and application addressing based on port numbering.

Internet Layer

It is located under the transport layer. This layer uses the Internet Protocol (IP) to carry data from the source to the destination endpoint. The services offered are host identification, based on IP addresses and interfaces, and packet routing from a source to a destination in a connectionless mode.

Link Layer

It is located on the bottom of the architecture. The link layer is used to communicate directly to the network. The abstraction provided to upper layers guarantees TCP/IP operations on virtually any hardware networking technology. It is sometimes referred to as the media-access layer protocol. Framing and data forwarding are the main operations performed by this layer.

Figure 2.1 and Figure 2.2 represent the layer relationship between the OSI and the TCP/IP model as well as a general overview of the distribution of protocols in the Internet Protocol Suite.

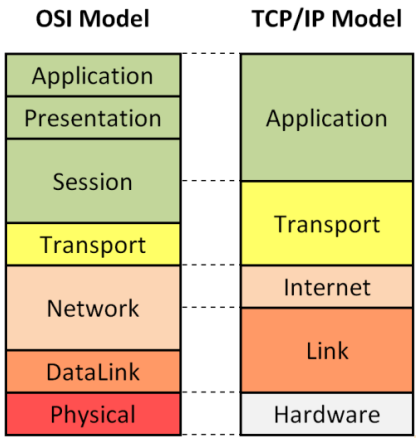


FIGURE 2.1 OSI MODEL AND TCP/IP MODEL

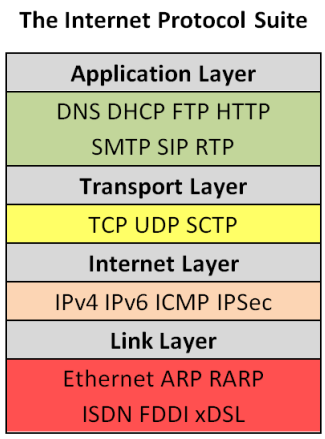


FIGURE 2.2 INTERNET PROTOCOL SUITE

2.2 Internet Protocol

The Internet Protocol was originally developed for the Department of Defense (DoD) at the University of Southern California. The protocol was first specified in 1980 and

described as "designed for use in interconnected systems of packet-switched computer communication networks" [10]. The first major release occurred in September 1981 when the version 4 of the IP protocol was released.

IPv4 establishes a framework for packet delivery from a source to a destination over a path of interconnected networks. The host identification in IPv4 is defined by fixed-length IP addresses of 32 bits. The IPv4 address space is able to allocate up to 2^{32} roughly 4.3×10^9 IP addresses. The mechanisms for reliability, sequencing as well as the flow and congestion control are typically provided by the transport protocols in the layer above.

Because the protocol is based on connectionless packets, on event of a network failure, IP uses Internet Control Message Protocol (ICMP) [22] to report an error to the originator of the packet. ICMP messages are sent in IP datagrams as if they were a higher level protocol, but it is considered to be a network protocol and an integral part of IP. The reliability in IP is not granted by ICMP since the primary purpose of ICMP is to provide feedback about an unsound situation.

ICMP operations are defined by a combination of type and code fields. The most common ICMP messages are:

- Echo request/reply: Originally intended to test the IP reachability of a given host, these messages are mostly used by the *ping* application.
- Destination unreachable: Error message indicating either network or host *failure*. The most common cases are network unreachable, host unreachable, port unreachable or fragmentation required.
- Time exceeded: Error message generated by an intermediary router that received a packet with TTL value 1. It is a very common error in scenarios where network loops are present.

IPv4 has suffered several modifications through the years as new functionalities have been implemented. Among them, it is worth mentioning differentiated services [19], congestion notification [24] and security features.

In the beginning, the network-addressing architecture consisted of *classful networks* that divided the address space into 5 different classes.

- Class A: Reserved for unicast use, allocates 8 bits for network addressing and 24 bits for hosts. Address space from 0.0.0.0 to 127.255.255.255.
- Class B: Reserved for unicast use, allocates 16 bit for network addressing and 16 bits for hosts. Address space from 128.0.0.0 to 191.255.255.255.
- Class C: Reserved for unicast use, allocates 24 bit for network addressing and 8 bits for hosts. Address space from 192.0.0.0 to 223.255.255.255.
- Class D: Reserved for multicast use. Address space from 224.0.0.0 to 239.255.255.255.
- Class E: Reserved for experimental use. Address space from 240.0.0.0 to 255.255.255.255.

Address exhaustion problem

The dramatic growth of the Internet in late 80s brought under the spotlight the limitations in terms of addressing and efficiency of the protocol. The major drawback of the *classful network* model was that many corporations required a larger addressing space than the *Class C* block provided. Instead, a *Class B* block had to be allocated, which, in most cases, was way larger than required. Consequently, the Class B pool was rapidly depleted because of the fast growth of the Internet. Over the following years, new techniques were developed to solve the problem of the address exhaustion.

CIDR

The adoption of Classless Inter-Domain Routing (CIDR) [25] in 1993 brought immediate benefits to both manufactures and researchers. CIDR is based on *variable subnet length mask* which allows a network to be divided into different-sized networks. This method allowed slowing the growth and reducing the size of the already burdensome routing tables on the routers across the Internet by performing route aggregation. In addition to improved and more efficient routing algorithms, it also alleviated the address exhaustion by enabling efficient mechanisms for address allocation.

Private networks and NAT

Nevertheless, the Internet continued to grow at a pace that concerned the community about an inevitable problem of the address exhaustion. In March 1994, the IETF brought to IP the concept of *public* and *private* realms [26]. This decision was driven by the necessity of a host to connect to either private or public services. As a

consequence, the IANA reserved three blocks of IP addresses for the private networks to satisfy these requirements and avoid unnecessary public address allocation. This new model benefited large corporations by expanding the address space available, that otherwise, should have been obtained from the public pool. The new address pools defined were the following.

- 10/8 prefix: Address space from 10.0.0.0 to 10.255.255.255.
- 172.16/12 prefix: Address space from 172.16.0.0 to 172.31.255.255.
- 192.168/16 prefix: Address space from 192.168.0.0 to 192.168.255.255.

Two months later, in May 1994, the concept of NAT was introduced [31]. Initially, NAT provided a basic form of address translation between realms. New versions of NAT have also implemented port translation; these are known as Network Address and Port Translation (NAPT). NATs have been used traditionally to connect isolated private networks with the Internet by sharing a public IP address. The main advantage of NAT is that it does not require any changes in either the hosts or the routers so they can be transparently added to the network. Moreover, NAT introduces an additional layer of security to the network by blocking new incoming connections towards the private hosts thus protecting them from public attacks. Ultimately, this functionality will result in the previously mentioned *reachability problem*. A set of protocols and techniques have been developed to solve this issue, they are classified under the name of NAT Traversal Protocols.

Figure 2.3 represents a scenario where a *private host* and a *remote host* communicate through a *NAT* device. It includes also the terminology that is used later.

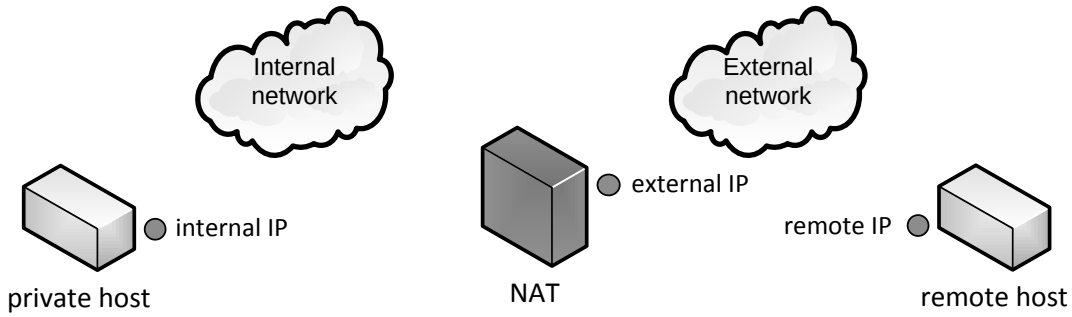


FIGURE 2.3 NAT ARCHITECTURE

When processing outgoing connections from a *private host* to a *remote host*, *NAT* creates a translation mapping from *internal* IP:port to *external* IP:port. Attending to this mapping, the RFC 4787 [3] has defined the following behavior:

- Endpoint-Independent Mapping: Reuses the existing mapping for the private host regardless of the remote host.
- Address-Dependent Mapping: Reuses the existing mapping for the private host regardless of the port in the remote host.
- Address and Port-Dependent Mapping: Reuses the existing mapping for the private host to the remote host while the mapping is still alive.

Additional policies have been defined regarding the allocation of external IP addresses and port assignments.

- IP address Pooling Mapping: Uses different external addresses when the private host establishes multiple sessions.
- Port assignment: Some NATs attempt to preserve the internal port number when creating a new mapping, this is called *port preservation*. In case of collision, a NAT may override a previous mapping, assign a new external address (if available), pick a new external port or perform *port overloading*.

Hereafter we explain the different behaviors of NATs based on an incoming connection originated in the *remote host* and the existence of mapping in the NAT binding *external* IP:port to *internal* IP:port.

- Endpoint-Independent Filtering: Accepts a packet that matches the existing mapping regardless of the *remote host*.
- Address-Dependent Filtering: Accepts a packet that matches the existing mapping regardless of the port in the *remote host*.
- Address and Port-Dependent Filtering: Accepts a packet that matches the existing mapping *if and only if* the *private hos*t has already sent packets to that particular *remote host* and *remote port*.

Internet Protocol version 6 - IPv6

In 1998 the IETF released the version 6 of the Internet Protocol, IPv6 [5]. The main advantage of IPv6 versus IPv4 is the extended address space supporting fixed-length addresses of 128 bits represented in hexadecimal notation. The addressing space in IPv6 is roughly 3.4×10^{38} IP addresses. Moreover, IPv6 implements new multicasting

mechanisms, stateless configuration for host auto-configuration and native support for IPsec among others. Consequently, a new version of the control protocol ICMP had to be developed, ICMPv6.

The transition from IPv4 to IPv6 requires modification of hardware and software in the devices connected to the network, and these are not exempt of problems. Despite the techniques developed to ease this transition such as *Dual Stack*, *IPv4 to IPv6 Translation*, or *IPv4 Tunneling* the slow penetration of IPv6 still raises some questions nowadays.

2.3 Transport Protocols

The transport layer is located between the Application and the Internet layer in the TCP/IP architecture model. Based on the RFC 1122 [4], "The transport layer provides end-to-end communication services for applications". These services are offered mainly by Transmission Control Protocol (TCP) and User Datagram Protocol (UDP). The new Stream Control Transmission Protocol (SCTP) is not yet widely deployed or natively supported by most operating systems. Due to these protocols being well known, only the most important features will be presented in this section.

TCP is a connection-oriented protocol that provides reliable data transfer and requires a connection setup. TCP implements mechanisms for establishing (three-way handshake), maintaining and tearing down a connection. The Protocol Data Unit (PDU) in the protocol is called TCP segment. TCP uses 16 bits length port numbers to identify the application end-points on a device. The main features provided by TCP are as follows:

- Stream-oriented: Transport chunks of information and creates an abstraction of continuous data flow between end-hosts.
- Ordered data transfer: Based on a sequence number.
- Packet retransmission: Based on acknowledgements of the received segments. Retransmissions are triggered upon timeout expiration.
- Error detection: Based on a checksum field contained in the header.

- Flow control: Avoids the host overflow by using a sliding window to determine the remaining amount of bytes allowed to be sent.
- Congestion control: Achieves a high data throughput and prevents network congestion collapse. The most popular algorithms are slow-start, congestion avoidance, fast-retransmit and fast-recovery. [2]

TCP is designed to perform general operations of reliable data transfer with no special constraints for timely delivery. Some common applications that use TCP are SSH, FTP, HTTP and email clients. On the other hand, TCP is not suitable for real-time applications and does not support broadcast or multicast traffic.

On the other hand, UDP is a connectionless message-oriented protocol [23]. Similarly to IP, UDP does not implement reliability or guaranteed ordering of packets. As a consequence, the applications are ultimately responsible for the retransmission and reordering the packets. The PDU in the protocol is called *UDP datagram*. Likewise TCP, UDP also uses 16 bits length port numbers to identify application end-points.

UDP is designed to accommodate real-time application requirements, being more efficient for transmitting small amounts of information due to a reduced header size compared to TCP. UDP also provides support for broadcast and multicast traffic. The protocol is mainly used by DHCP, DNS, TFTP and VoIP applications.

2.4 NAT Traversal Protocols

The adoption of NATs due to the address exhaustion introduced the reachability problem. This is an important issue that strongly requires a solution. There have been many proposals throughout the years about how to solve this problem. These protocols and techniques have been classified under NAT Traversal Protocols. Among them, STUN/TURN/ICE is the recommended solution by the IETF.

STUN - Session Traversal Utilities for NAT

STUN does not actually provide a solution to traverse NATs by itself but instead, it helps to determine if an endpoint is currently located behind a NAT [31]. STUN is

also able to detect the types of mapping and filtering behavior of a NAT. In addition, it can be utilized to check connectivity between two endpoints and as a keep-alive protocol to maintain the existing binding in NATs. On the other hand, it does not help if both endpoints are behind different NATs that use a strict filtering policy.

The operation requires at least a STUN server in the public network listening to two different IP addresses, or two STUN servers for that matter. The private host behind the NAT will send binding requests to the servers, which respond indicating the public IP address and port number allocated by the NAT within the STUN protocol data. By comparing these results the host learns the outgoing policy in NAT. The filtering behavior is also tested by sending incoming requests from the STUN servers to the private host and analyzing what kind of connections are indeed allowed by the NAT.

TURN - Traversal Using Relays around NAT

TURN is an extension to the STUN protocol that enables a host located behind a NAT to receive incoming TCP or UDP connections by relaying the traffic through a public TURN server [28]. TURN introduces new and extended signaling in order to distinguish STUN requests from user data and additional framing to separate different connections. Opposite to STUN, TURN enables two endpoints located behind different NATs to communicate with each other by relaying the information through the TURN server. In the end, TURN achieves connectivity despite the strictest filtering policy of NATs.

In terms of operation, a host behind a NAT may set up a session with a public TURN server specifying the destination endpoint and forwarding rules. The TURN server stores session state per client. The private host is responsible for adapting the packet information in such a way that the responses from the destination hosts are sent to the relay server. The TURN server ultimately forwards the received data to the TURN client through the original connection traversing the NAT. The addition of a relay element may affect the quality of the connection by introducing packet loss as well as increasing the end-to-end delay and latency. Consequently, a TURN server constitutes a single point of failure that may affect the availability and stability of the system.

ICE - Interactive Connectivity Establishment

ICE introduces new operations to determine the best path between two endpoints, despite the existence of NATs, combining the functionality provided by STUN and TURN [17]. ICE is able to successfully establish a connection even under very challenging network conditions. There is currently some research to adapt TCP into ICE, but the main focus is still UDP-based multimedia sessions based on offer/answer model such as SIP/SDP.

ICE operates on the client by detecting endpoints potentially reachable by the remote host. This information is collected from the local interfaces, point-to-point links if available as well as STUN and TURN operations. The information is added to the SDP content and tested with the STUN protocol to detect connectivity between clients. End devices use both STUN and TURN protocols in order to achieve end-to-end connectivity and forward the corresponding data or media flows.

Hereafter we list some facts about ICE:

- Provides dynamic discovery of the shortest path between end-points.
- Works through virtually any kind of NAT/Firewall device but it does not necessarily require the endpoint to discover NATs and behaviors.
- Guarantees that a media connection is already established before the device alerts of an incoming request, otherwise the endpoint never *rings*.
- Usage of relays is limited to the worst case scenario where no other possibility is available, usually when both endpoints are behind an address and port-dependent NAT.
- Introduces considerable delay to session setup.

In spite of being the recommended solution by the IETF, the STUN/TURN/ICE solution also include some drawbacks that hinder its adoption, especially in conjunction with mobile devices.

- STUN/TURN/ICE forces a mobile device to wake-up for the keep-alive signaling preventing a binding from expiration; usually this has to be done per each application that wishes to be reachable from the Internet. As a consequence, the battery of the mobile device is quickly depleted.

- STUN/TURN/ICE client side code needs to be integrated within each application making application design more complex and memory hungry.
- STUN/TURN/ICE introduces a significant delay in session setup of communications applications. This is because the goal is to find the optimal configuration among the candidate addresses and the application must wait for expiration timeouts before some of the options can be discarded.

ALG - Application Layer Gateway

The main purpose for an Application Layer Gateway is to enable end-to-end connectivity between hosts when the connection traverses a NAT or firewall and the address realms are naturally different. These operations are traditionally performed by the NAT device and triggered when a packet traversing the device matches a rule. The ALG may interact with the forwarding table by setting a new mapping or modifying the content embedded in the packet to adapt between address realms.

The major disadvantages of the ALGs are that they are completely application/protocol specific and utterly dependent of the NAT implementation. Consequently, modern firewalls are stateful and routinely use NAT in conjunction with ALGs for many protocols such as FTP, H.323 and SIP.

On the other hand, newer applications are usually NAT-friendly, thus they do not require such operations.

UPnP - Universal Plug and Play

UPnP provides mechanisms for seamless device discovery and communication between computers, access points, printers or gateways [33]. The main concept behind "*plug-and-play*" consists of enabling auto-configuration and ready-to-use functionality when the device is plugged into the network. UPnP is a standard promoted by ISO/IEC that was first released back in 2008.

The operation in UPnP is based on a distributed open architecture following the Internet Protocol Suite model relying on SSDP, HTTP, XML and SOAP at the application layer. Regarding NATs, a new protocol Internet Gateway Device (IGD

Protocol) uses UPnP to establish a communication with a gateway. This session allows to create, modify and delete bindings dynamically thus enabling incoming connections through the NAT device towards the private network. Because UPnP lacks authentication mechanisms it is usually disabled by default for security reasons.

Regarding the operation of NATs, it is worth mentioning the roles of the timeout and the keep-alive. When an entry is added to the forwarding table, a timeout is dynamically generated for that particular entry and its value depends on the protocol in use. The timeout is used to indicate the last time an entry was used. If the timeout expires, the NAT can delete the mapping from the forwarding table thus freeing the allocated resources. Due to these features, most of the techniques described in this section make use of keep-alive methods to prevent the mapping from expiring by sending *any* information through the NAT extending the timeout.

2.5 Domain Name System (DNS)

This section introduces the main concepts of the Domain Name System (DNS). First the history and motivation of the protocol are described. Then, the architecture is explained followed by an operation example.

History and Motivation

Early in 1980s, a host device stored a file on the local file system with the name and address of the remote device they wanted to connect to. New destination hosts had to be manually appended to the file. As a result of the Internet growth, the model was no longer scalable and DNS was first developed in 1983.

DNS is the naming system used in the Internet nowadays. It was originally conceived to provide translation of domain names (in ASCII characters) to IP addresses, considering the difficulty for human beings to remember IP addresses. The protocol has suffered severe variations and new functionalities have been added ever since.

Architecture

DNS relies on a distributed and hierarchical architecture of interconnected name servers. Data stored in a name server becomes virtually available everywhere following a client-server solution. There are three major components within the DNS architecture. [1]

Domain name space

The domain name space consists of a distributed database following a tree in a hierarchical fashion similar to the Unix file system. These databases are often referred as *zones*. The tree contains a single root and is extended by *subdomain* names. Each of these domains may contain several subdomains as well, resembling a branch in a tree structure. Subsequently, data stored in a domain receives the name of resource record similarly to the leaf of the tree. A resource record consists of a tuple of information that contains the following fields: name, type, class and TTL. Although the complete list for DNS type of records is publicly available, hereafter we mention the most common ones.

- A: Maps a hostname to an IPv4 address of the host.
- AAAA: Maps a hostname to an IPv6 address of the host.
- PTR: Maps an IPv4 or IPv6 address to a hostname.
- SOA: Indicates the start of a zone of authority.
- NS: Indicates the authoritative name server for a delegated zone.
- CNAME: A canonical name for an alias.
- MX: Maps a domain name to a mail exchange server for the given domain.
- NAPTR: Naming authority pointer that allows expressions encoded as URIs.
- SRV: Maps services and transport protocols to other domain names and port numbers. Only supported by a few applications.

Name servers

A name server implements the server role in the client/server DNS architecture. A name server is a node that contains information about a zone of the domain name space. Name servers generally contain complete information about the zones they control. There are two types of name servers in DNS. The primary master servers

(master) load the zone information from a *zone datafile* located in the local file system. The secondary master servers (slave) download a copy of the zone file from the master server. This operation eases the maintenance of the zone data files and implements redundancy by having the same data replicated in several servers.

Resolvers

A resolver implements the client role in the client/server DNS architecture. The operations performed by the resolver are the following:

- Querying a name server about certain name and record type.
- Interpret the response obtained which may contain a record or an error.
- Return the information to the application that requested it.

Operation

The process where a resolver queries a name server and retrieves data about a particular domain is called *name resolution*. Because of the actual structure of an inverted tree, the resolution process is always initiated from top to bottom. The resolution starts by contacting the top level domain, stepping one level down at a time following referrals until the given resource record is found. There are two modes of operation defined in DNS, recursive and iterative.

Recursion

In this operation mode, a resolver delegates most of the burden of the resolution to the name server that processes the query. This is often the case where the resolver does not have the intelligence or resources to follow a referral and perform the whole resolution on its own. In this case, the queried name server is obliged to respond with either the requested data or an error message. If the server does not contain information about the domain, it must follow the referrals until an answer is found.

Iteration

This operation is much lighter compared to the recursion. Upon receiving an iterative query, the name server only needs to return the best answer that it already knows without engaging in any other querying operation. The server consults its local information and gives either the record requested or a referral to the next name server in the tree.

Example

The example depicted in Figure 2.4 represents a name resolution for the domain *www.aalto.fi* issued by *host*. The process starts with *host* sending a recursive query to *dns.foo*. Without information about the domain, the server contacts the *Root-DNS*. Consequently, iterative queries are issued between name servers and referrals are followed until the resource is found and forwarded to the originating *host*.

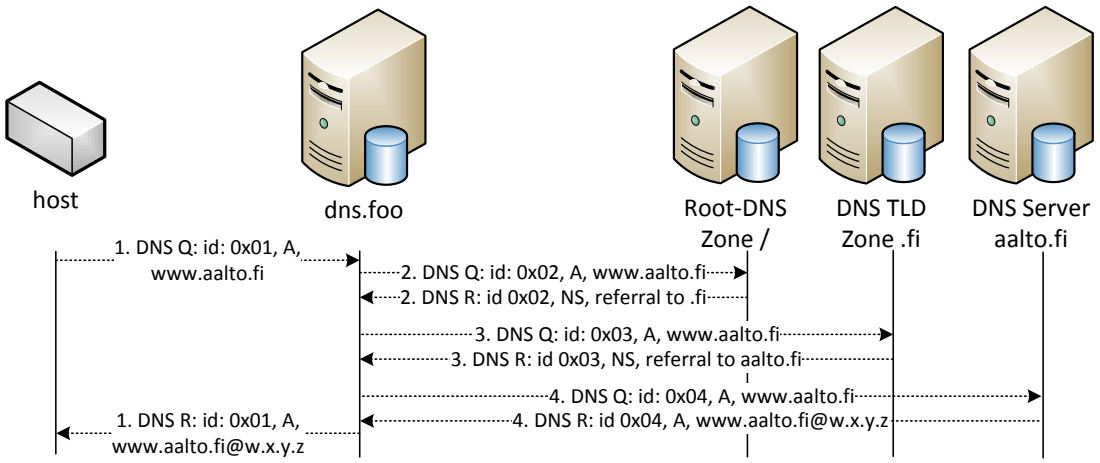


FIGURE 2.4 DNS DOMAIN RESOLUTION EXAMPLE

3. Customer Edge Switching

The purpose of this chapter is to introduce the proposed solution of the Customer Edge Switching. The chapter first describes the motivation and research background. Then, the system architecture is discussed. Finally the implemented solution is presented together with the different scenarios for data forwarding.

3.1 Motivation

Back in 2009 Lauri Virtanen presented his Master's Thesis "Communicating Globally Using Private IP Addresses" [34], driven by the paradigm of the reachability problem present with NAT deployments. The study also considers the necessity of using IPv4 as public identifiers and whether other technologies such as IPv6 or Ethernet could replace the core network. This would create an abstraction layer establishing different realms of technologies, transparent to the user. A network prototype was created, continuing the previous work initiated by Raimo Kantola on Future Internet [14].

Customer Edge Switching [13] aims at placing hosts in private networks enabling end-to-end connectivity addressing the reachability issue as well as improved security by using *Customer Edge Traversal Protocol* (CETP) [12] with Customer Edge Switches. Ultimately, the goal is to replace NAT devices implementing the benefits of NATs with additional enhanced capabilities such as trust and smart inbound traffic management. CES is a way of moving from the end-to-end principle to the trust-to-trust principle advocated by David Clark.

The CETP is an edge to edge protocol for tunneling packets between Customer Edge Switches in different networks that each have their own private address space. The protocol provides identification of the communicating hosts by carrying IDs and establishes a dynamic tunnel for forwarding data between the devices. As of September of 2012, CETP is still a work in progress being developed by Maryam Pahlevan at the Department of Communications and Networking of Aalto University.

3.2 Architecture

CES divides the global network into two areas, the User Network (UN) and the Service Provider Network (SPN). End users are connected to a UN; each of these UNs contains at least a CES device. UNs are independent and isolated from each other resulting in no direct communication. The CES device has at least two interfaces that connect to the UN and the SPN networks. Respectively CES also provides firewall and gateway functionalities in addition to a large pool of private IP addresses for the UN, onwards referred to as *proxy-addresses*.

The separation of UN from SPN has the benefits of isolation and transparency. Considering the SPN is completely operator dependent, it becomes possible to deploy new protocols and technologies in such networks thus leading to enhanced performance and improved capabilities for packet forwarding. For example, a core network could be running IPv4, IPv6, IP/MPLS or Ethernet independently from the technology used in the UN. In addition, the core network provides Directory Services (DS) for domain resolution such as DNS.

The users, the hosts and the services are identified with IDs. These IDs can be randomly generated by the CES device based on its own algorithms or retrieved from an operator as a derivative of any reference of the customer. The remote users are dynamically represented to the private hosts by allocating a *proxy-address* from the available pool and with additional state information in the CES device that links the originating and the recipient hosts. The CES architecture is represented in Figure 3.1.

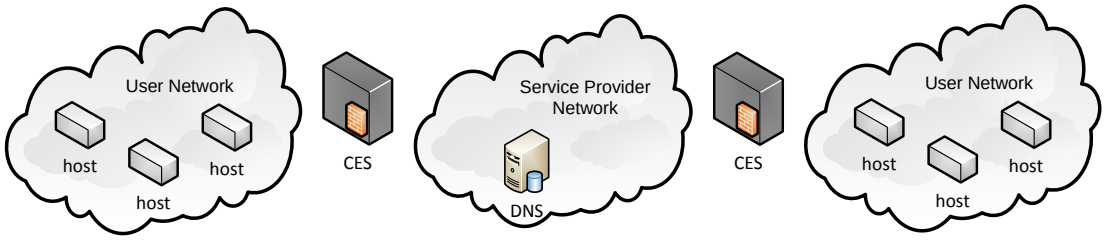


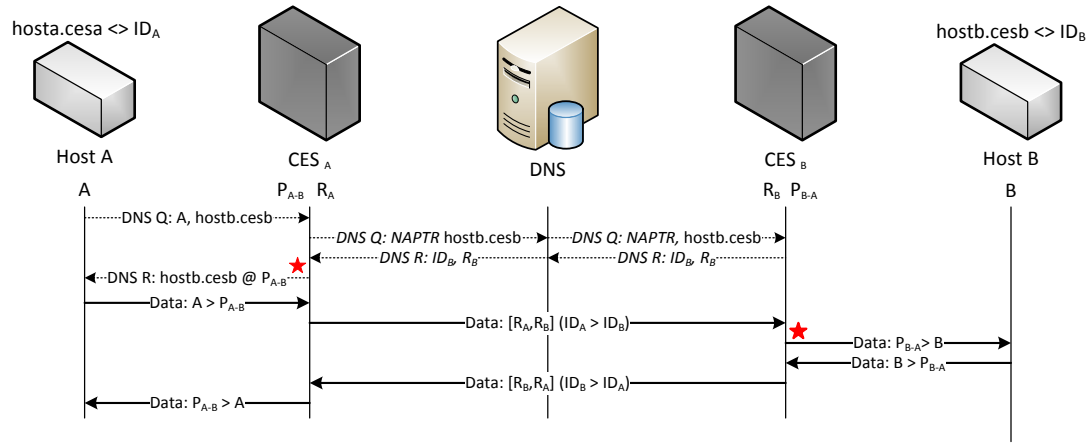
FIGURE 3.1 CUSTOMER EDGE SWITCHING - ARCHITECTURE

3.3 Packet Forwarding in CES

The philosophy of the CES concept lies in the domain name providing identification of the users, host and services. The result is that IP addresses can no longer be used to that end. Each of the User Networks has its own addressing enabling the separation of IP addresses and names. CES communications rely heavily on domain name queries to create a well-defined state with host information and the chosen virtual anchor (proxy-address) for the subsequent forwarding of data packets. A domain resolution operation must be always placed first in order to create a valid state in CES, obtain a proxy-address and subsequently forward the data packets.

DNS implements multiple types of records. From the variety of record types previously covered in Section 2.5, CES uses NATPR type to communicate with another CES. In contrast with traditional A or AAAA records, NAPTR provides extensibility and better support for abstract identifiers.

Figure 3.2 introduces a case example of CES communication with the corresponding packet flow. The scenario consists of two private hosts, *Host-A* and *Host-B*, located in different UNs, two CES devices, CES_A and CES_B, connecting both UNs through the SPN and a DNS server located in the SPN as well. The addressing used is IPv4.



A: Private IP of Host-A	ID_A: ID of Host-A	R_A: Routing locator of CES_A
B: Private IP of Host-B	ID_B: ID of Host-B	R_B: Routing locator of CES_B
P_{A-B}: Proxy-address representing Host-B to Host-A		
P_{B-A}: Proxy-address representing Host-A to Host-B		

FIGURE 3.2 CES TO CES COMMUNICATION FLOW

The following lines explain the operations performed on each side of the communication, originating and recipient side, according to the previous figure. It is

also worth mentioning the red star mark that corresponds with the creation of state in CES.

Originating side - CES_A

Originating host *Host-A* attempts to establish a connection with a server reachable via its FQDN *hostb.cesb*. Host-A sends a DNS query to CES_A that processes it and initiates a NAPTR type resolution for the given domain. The DNS response conveys information about the remote host ID and the routing locator for the remote CES_B in the SPN network. Both CES devices are visible to each other via the SPN and the locators used are according to the technology used by the network. The state information is dynamically created to allow the forwarding of data packets. Then, CES_A sends the DNS response to Host-A, containing an allocated IP address from its private pool of addresses. As a result, Host-A sends its IP packets to the given address where CES_A will process and forward them accordingly towards the remote CES_B.

Although the encapsulation used in the core network is out of the scope of this chapter it is worth indicating that the IDs from both hosts involved in the communication as well as the routing locators of both CES are carried in the packet.

Recipient side - CES_B

The process starts with an incoming DNS query requesting a NAPTR record. A DNS response is generated with the host ID and the local routing locator of the CES_B in the SPN network. Afterwards, an incoming data packet is received in CES_B originating from CES_A. The host IDs are extracted from the packet and a new proxy-address is allocated for this communication. State is created with the IDs of the hosts, the remote CES_A and the proxy and private addresses for the given host. The IP header is rewritten with the new information and the packet is finally delivered to Host-B.

Summary

The previous process demonstrates how clients located in private realms are able to communicate with servers in different private realms, thereby solving the reachability

problem. Furthermore, end users are unaware of core network technologies and completely independent from the operations performed to facilitate the end-to-end communication. In addition, the process of forwarding DNS NAPTR queries, allocating proxy-addresses and creating state in the CES appears to be completely transparent to the user.

With regard to the name resolution, an incoming request in the CES device is just designer's choice. Due to the fact that incoming DNS queries are not necessary for state allocation the name resolution operations can be offloaded to any DNS server.

3.4 Evaluation

This section focuses on briefly describing the results upon submitting to test the network prototype implemented by Virtanen. Afterwards, the advantages and disadvantages of the concept are discussed.

Basic TCP / UDP operation

The prototype seems to work properly with basic TCP and UDP flows taking place between *Host-A* and *Host-B*. Applications such as SSH, telnet or netcat work successfully. On the other hand, some issues were detected while using file transfer over SSH connections or downloading certain content from a web server. The problem is just an implementation issue due to the overhead introduced by CES and the fragmentation of IP packets.

Basic ICMP operation

Testing with ping application from Host-A to Host-B revealed that the packets flow properly but with a certain delay between requests due to DNS PTR queries that are not processed by the CES device. On the other hand, ICMP error messages are not properly handled because they contain inner IP headers with IP addresses that belong to the remote realm address instead of the local one.

FTP Application

Testing the prototype with FTP protocol fails. FTP uses a TCP control connection for signaling information and creates additional TCP connections for the data transfers. The parameters for this data transfer are sent within the control connection and indicate the IP address and port number where the data connection should be established. Taking into account that the payload information is never modified, the data transfer fails due to incorrect information on the application layer.

SIP Application

Testing the prototype with SIP protocol fails. As it previously happened with FTP, SIP also makes use of IP addresses on the messages exchanged on the application layer. Although both SIP and SDP semantics support domain names to identify hosts, the applications tested usually perform a domain resolution for the given domain therefore conveying IP address on the application layer.

Summary of testing

There are certain scenarios where faulty operations arise due to the architecture and design. The majority of the problems seem to be related to the packet forwarding functionality. Whereas the CES concept appears to provide abstraction towards transport layer protocols, the reality is that in practice the user data has to be modified. As a result, the difficulty lies in adapting the scope of the user data to a proper value on the remote side thus requiring additional and specific operations for several protocols.

On the other hand, the rest of the protocols and applications used do not appear to be affected at all. Despite further testing is still pending it seems accurate to state that the CES architecture enables smooth end-to-end communication between hosts insofar "NAT friendly" protocols and applications are in use.

It is important to clarify that Virtanen did not study the interworking of CES with legacy customer networks but rather the communication between two different CES networks.

4. Requirements and Design Objectives

The term *communication* itself implies the transfer of messages to others. In that sense, the primary goal is to overcome the limitations introduced by NATs by enabling end-to-end connectivity and solving the reachability problem. This chapter focuses on establishing solid foundations and requirements attending to functional requirements and design objectives.

4.1 Connectivity Requirements

The proposed solution has to enable end-to-end communication with the existing protocols and applications. The communication must flow undisrupted for the supported transport protocols: TCP, UDP and ICMP. The solution has to support most common applications such as email, DNS, HTTP, FTP, SSH, SIP or Skype.

4.2 Flexibility Requirements

The model has to enable support to new protocols or applications. In addition, it should be able to interwork with firewalls and NAT devices that could exist between the end devices. We will develop new Application Layer Gateways (ALGs) for those protocols that may result in *broken* or *defective* communication because of the NATs. In most cases, protocols that are not "*NAT friendly*" fail to operate at all or they work with certain restrictions.

4.3 Scalability Requirements

The goal is to model an architecture that by making use of a limited number of public IP addresses is able to attend to the needs of a large number of users. The solution cannot aggravate the existing problem of address exhaustion. In addition, the marginal

cost of adding a new host to the system must be balanced with the necessity of allocating more IP addresses and the service offered to the rest of the hosts.

4.4 Deployment Requirements

The proposed model must provide a transparent framework without affecting the existing network elements and infrastructure. No modifications can be introduced to the current protocols, hosts or customer networks that have not invested into the CES/PRGW technology. In addition, the deployment has to be motivated by technological and economical factors that benefit the main players and promote adoption.

As a result, by providing the Private Realm Gateway functionality, it becomes possible to deploy Customer Edge Switching one network at a time. On the other hand, it is also desirable that the PRGW operations are independent from the CES architecture.

4.5 Security and Trust Requirements

The system has to avoid introducing points of failure vulnerable to DoS/DDoS that hinders the security of the platform. The model should also introduce additional security mechanisms that protect the hosts against public attacks or malicious users. Suspicious traffic or behaviors of any of the hosts involved in the communication is to be analyzed and reported so the adequate measures can be applied. In this sense, the system gradually moves towards *end-to-end trust*.

5. Connectivity Models in CES and Internet

This chapter focuses on the main differences extracted as a result of the comparison between the CES and the Internet models. The chapter aims to lay down some important principles and requirements, setting the foundations for a new design. The chapter first refers to the notation used. Then the different models attending to outgoing or incoming connections are analyzed. The scenarios illustrated in this chapter operate with IPv4 addresses.

5.1 Notation and Definitions

Because the following chapters make use of specific acronyms or terms, hereby we provide a definition in order to avoid misunderstandings. The following lines offer a brief explanation about the notation and symbols used in the figures in order to enhance readability and understanding

- NAT: Represents an operation of address translation adapting IP addresses and ports between private and public realms.
- Reverse NAT: Represents the reverse operation of a previous translation of IP addresses and ports between realms.
- NAT Table: Represents a database that keeps state of current connections mapping private, outbound and public IP addresses and ports.
- Socket: It is represented as: (IP_Address:Port_number)
- Connection: It is represented as two connected sockets a bidirectional communication between two hosts. (IP_ A:Port_A) > (IP_B:Port_B)
- Realm: In order to differentiate between private and public realms, the suffixes “i” for inward and “o” for outward are prefixed to a particular element, either IP address or port number.

5.2 Outgoing Connections

This section explains how a host located behind a NAT device is able to initiate a communication with another host located in a public realm network.

5.2.1 Overview

The CES model relies heavily on DNS queries to create a well-defined soft-state and forward consequently meaningful data between the end hosts. An overview of the operational mode has been already discussed in Chapter 3 thus only some particularities will be studied here in detail. As a result, the hosts behind a CES device must perform a DNS request in order to retrieve the resulting *proxy-address* where the data will be forwarded to. Moreover, in the Internet model, DNS queries are greatly used for web browers or mail exchangers among others. In light of these facts, it seems feasible that both models could benefit to some extent from the same operations.

However, there are more scenarios than the previously discussed thus the next question arises: *"What happens if no DNS lookup process is involved?"*

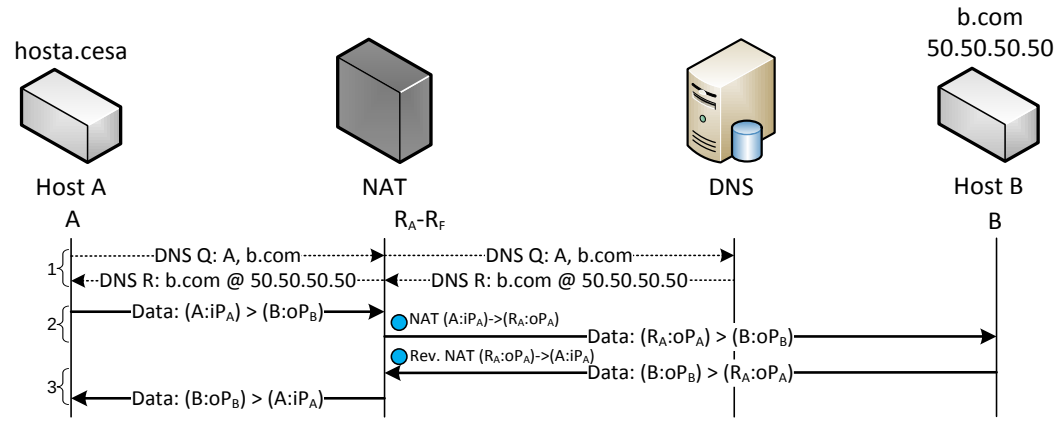
In the CES model, an allocated proxy-address is mapped to the private host representing the remote host, having zero or no valuable meaning outside that particular scope. A connection towards a non-allocated proxy-address results in packet drop by the local CES because no forwarding information exists for that communication.

On the other hand, in the Internet model a host can directly send IP packets to another host without previous DNS lookup process. It is very common for applications to use fixed IP addresses for initialization or setting up a connection. Therefore, it does not seem feasible that the CES-to-CES model with the proxy allocation policy could be applied under these circumstances.

Needless to say, the new model has to be compatible with both operations regarding name resolution so the utmost compatibility and transparency can be achieved.

5.2.2 Scenario example

The following scenario consists of a host located behind a NAT device, attempting to connect with another host in the public network. The process is depicted in Figure 5.1 and represents an outgoing communication following the Internet model while traversing a NAT device. The operation uses domain resolution prior to sending meaningful data and therefore it represents the most *complex* model for legacy communications. In addition, consider a pool of public IP addresses (R_A-R_F) on the public interface of the NAT and no explicit allocation policy for these addresses.



A: Private IP of Host-A	iP_A: Local port of Host-A	oP_A: Public port of Host-A
B: Public IP of Host-B	oP_B: Local/public port of Host-B	R_A: Public IP for Host-A

FIGURE 5.1 INTERNET: OUTGOING CONNECTION

#1 Originating Hosta-A, aware of the FQDN of Host-B, sends a DNS query for the domain *b.com*. The NAT forwards the request to the DNS server and obtains a response. The response is sent back to host-A with the IP address of the remote host – 50.50.50.50.

#2 Host-A creates a local socket with Host-B's information, $(A:iP_A) > (B:oP_B)$. When the packets traverse the NAT, it performs a private-to-public address translation for the source IP address and the port. The translation maps $(A:iP_A) \rightarrow (R_A:oP_A)$ and the entry is added to the forwarding table. Then the packet is forwarded to Host-B.

#3 Host-B receives the packet and creates the socket $(B:oP_B) > (R_A:oP_A)$, where $(R_A:oP_A)$ is the public identifier for Host-A. The responses are sent back to the NAT device. A query in the forwarding table matches an existing entry and obtains the mapping $(R_A:oP_A) \rightarrow (A:iP_A)$. The NAT performs a public-to-

private translation for the destination IP address and the port. Then the packet is forwarded to Host-A.

Note that the domain resolution does not affect the communication considering that a host could directly initiate a connection knowing the IP address.

5.2.3 Conclusions

The scenario explained previously represents how a private host using name resolution can initiate a communication with a public host on the Internet traversing the local NAT.

An attempt to establish a connection by using the same method as in the CES model reveals that allocating a proxy-address for representing a public host collides completely with the scenario of a host connecting directly to a particular IP address. In addition, applications that infer information based on the nature of the destination address realm, such as private or public, may result in misbehavior when for example, policies for bandwidth allocation are applied. As a result, a network scenario with mixed connections using both public IP and proxy IP addresses towards the same host must be avoided by all means in order to prevent undesired effects.

The model selected for handling outgoing connections resembles the traditional NAT behavior providing seamless mechanisms adapting between different realms of addresses regardless of name resolution.

5.3 Incoming Connections

This section explains how a host located in the public network is able to initiate a communication with another host that is located behind a NAT device and how these connections are able to traverse the remote NAT.

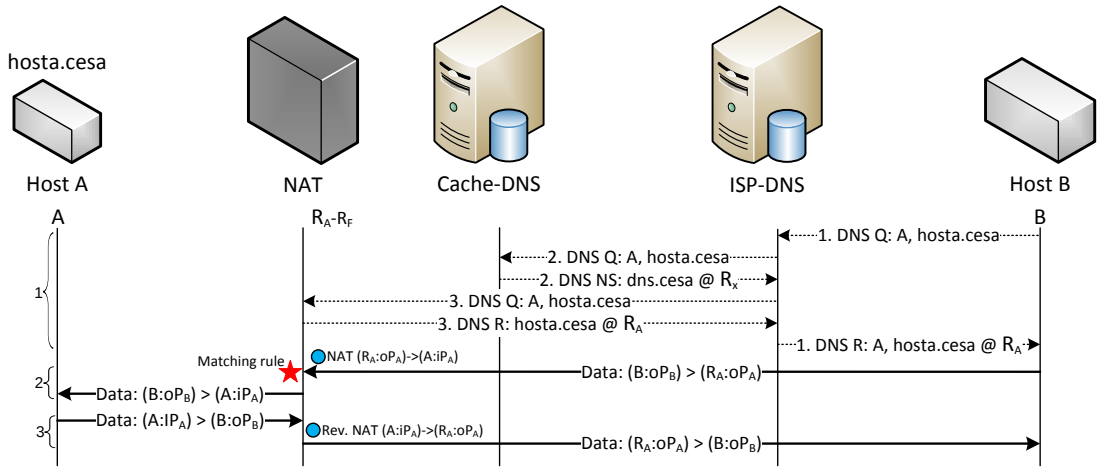
5.3.1 Overview

The CES model does not necessarily require incoming DNS queries to be received on the recipient side given the fact that soft-state is only created upon receiving the first data packet. The response to a NAPTR query therefore can be considered as mere routing information to indicate the originating CES how to build the packet.

However, the Internet model is somewhat different. DNS servers are traditionally located in the ISPs networks and provide name resolution for their customers. It is also usual that medium to large corporations have their own DNS servers managing their respective delegated zones. On the other hand, it is seldom that relatively small networks implement their own DNS server. For this reason, it is a common practice to register a domain name and have it redirected to a particular IP address. The resolution of such domains can be performed to either a fixed or a dynamic address updated via DDNS.

5.3.2 Scenario example

The following scenario consists of a host located in the public network attempting to connect with another host located behind a NAT. The process depicted in Figure 5.2 represents an incoming communication following the Internet model while traversing the NAT device. The operation uses domain resolution prior to sending meaningful data and therefore it represents the most *complex* model for legacy communications. Consider the NAT as the authoritative name server for the zone "*.cesa*".



A: Private IP of Host-A	iP_A: Local port of Host-A	oP_A: Public port of Host-A
B: Public IP of Host-B	oP_B: Local port of Host-B	R_A: Public IP for Host-A

FIGURE 5.2 INTERNET: INCOMING CONNECTION

#1 Originating Hosta-B, aware of the FQDN of Host-A, issues a name resolution for the domain *hosta.cesa*. ISP-DNS uses Cache-DNS to resolve the authoritative name server for the domain. The subsequent query received in NAT selects the R_A address from the pool as the response, which is ultimately forwarded to Host-B.

#2 Host-B creates a local socket with Host-A's information, $(B{:}oP_B) > (R_A{:}oP_A)$ and forwards the packet. Upon receiving the packet in NAT, it performs a public-to-private translation for destination IP address and port. The mapping is $(R_A{:}oP_A) \rightarrow (A{:}iP_A)$ and it *matches a predefined or existing rule*. A new entry is added to the forwarding table and the packet is forwarded to Host-A.

#3 Host-A creates a local socket with Host-B's information, $(A{:}iP_A) > (B{:}oP_B)$. When the packets traverse the NAT, the NAT device performs a private-to-public address translation for the source IP address and the port. The translation maps $(A{:}iP_A) \rightarrow (R_A{:}oP_A)$ and the packet is forwarded to Host-B.

Note that the domain resolution does not affect the communication considering that a host could directly initiate a connection knowing the IP address.

This scenario provides the following insight:

- Advantages: Awareness of name resolution of hosts connected to the private network. Possibility to create certain state with such information.
- Disadvantages: Additional complexity in NAT device and inability of forwarding packets that do not satisfy an existing rule or current forwarding information.

5.3.3 Conclusions

The scenario explained in the previous section represents how a public host on the Internet may attempt to initiate a connection with a private host traversing a remote NAT. It is very important to clarify that such operation only succeeds whenever there is an *existing entry in the forwarding table* or a fixed configuration for port forwarding. Both of these actions enable routing of packets towards the private network filtering by protocol and port numbers. A NAT receiving a packet for which

it does not satisfy any matching rule drops the packet, thus preventing an incoming connection from reaching its destination.

Despite it does not seem feasible to create state upon receiving a DNS request for data forwarding, our model could benefit from the awareness of the domain resolution regarding the private hosts. Additional scenarios were also studied regarding this matter and they have been included in Appendix A.

6. Design Candidates for Interworking

This chapter will introduce three design candidates for implementation within the current Internet architecture that attend to great extent to the requirements previously proposed. The purpose of this chapter is to explore the different possibilities in order to provide first and foremost connectivity followed by flexibility and scalability.

When designing these scenarios some questions arose on establishing the foundations for the interoperability model. These questions were the following:

- Is it possible to adapt the current CES operations to the Internet model?
- Is it possible to create forwarding state based upon DNS messages traversing the NAT device? For incoming traffic? For outgoing traffic?
- Is it possible to develop new efficient mechanisms for public address allocation yet enabling several hosts behind a NAT to offer the same services?

Because of the actual implementation will be developed together with the CES network prototype, future figures will refer to CES as the gateway of the private network. Due to the architecture, the new CES also has to resemble a NAT behavior providing address translation to adapt communications between different realms. The foremost kind of firewall is therefore provided.

6.1 Unique Global IP

The model allocates a public IP address per each one of the hosts located in the private network behind the CES device. The underlying idea in this design is mainly to overcome the reachability problem thus enabling maximum compatibility with the current applications. The following lines offer a brief explanation of the advantages and the disadvantages of the model attending the proposed requirements.

Advantages of the model

Connectivity: The reachability problem is solved allowing end-to-end connectivity by forwarding all the incoming traffic to a specific private host. The hosts in the

private network become reachable from the public realm. The address translation is static and transparent to both of the end hosts.

Flexibility: New protocols can be supported but still certain ALGs should be implemented to provide communication for protocols that are not NAT friendly.

Disadvantages of the model

Scalability: The design is highly demanding in terms of consumption of public IP addresses considering a ratio of 1:1 private to public. It does not contribute to alleviating the address exhaustion.

Deployment: The model can be easily integrated within the current network infrastructure but does not provide any technological or economical benefits for operators.

Security: End hosts are exposed to attacks originating in the Internet although firewalling techniques can be applied to prevent these attacks.

This model resembles a Demilitarized Zone (DMZ) behavior in NAT. The scenario is represented in previous Figure 5.1. Despite the fact that the deployment and security requirements are partially fulfilled, the design presents serious concerns in terms of scalability and the number of public IP addresses required for the private hosts.

6.2 Circular Pool of Public IP Addresses

This design allocates a fixed pool of public IP addresses on the CES device. These addresses are shared among the hosts located in the private network and *consumed* as they are needed thus enabling end-to-end connectivity. The challenge of this scenario lies on managing the incoming connections; applying policies for the address reservation and the forwarding of the packets to their final destination.

Advantages of the model

Flexibility: It requires the development of ALGs for the interworking with protocols and applications that are not NAT friendly.

Scalability: A limited consumption of public IP addresses may contribute to alleviating the address exhaustion.

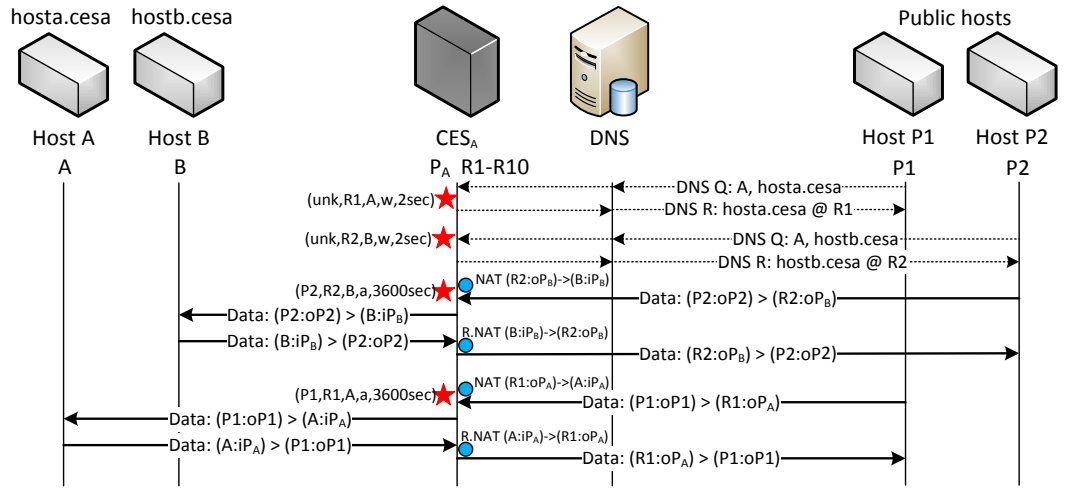
Deployment: Does not require changes in the current network infrastructure. The limited resource allocation required may bring economical benefits to operators.

Disadvantages of the model

Connectivity: A domain resolution operation is required on the remote host to allocate an IP address from the pool. A private host cannot be reachable directly by a public IP address since it is shared with other hosts.

Security: Although the private hosts are not exposed to attacks originating in the Internet, firewalling techniques can be applied to increase the security. On the other hand, an attacker could hijack the state created for another host resulting in denial of service from the originator perspective. Additional mechanism will have to be developed to mitigate these attacks.

This model requires incoming DNS queries to arrive to the CES device. The information conveyed in the query is then used to create a well defined soft-state in order to forward subsequent data packets to the queried host. Public addresses are reserved and consumed in an ordered fashion to assure connectivity to all hosts. The scenario is represented in Figure 6.1.



A: Private IP of Host-A	iP$_A$: Local port of Host-A	oP$_A$: Public port of Host-A
B: Private IP of Host-B	iP$_B$: Local port of Host-B	oP$_B$: Public port of Host-B
P1: Public IP of Host-P1	oP1: Public port of Host-P1	R1-10: Public pool of addresses allocated in CESA
P2: Public IP of Host-P2	oP2: Public port of Host-P2	
Tuple state definition (IP source, Public IP in CES, internal host, status(waiting/active), timeout)		

FIGURE 6.1 CIRCULAR POOL OF PUBLIC IP ADDRESSES

6.3 Domain Based Packet Forwarding

This design is specially designed for those protocols that use domain names in the user data. By performing operations as deep packet inspection it would be possible to retrieve specific information regarding the domain and the host names embedded in the payload. By cross-referencing these names with a database of users it would be possible to identify univocally a specific host thus enabling end-to-end connectivity for that communication.

Advantages of the model

Flexibility: If the communication uses a supported service, the identification of the end host indicates the destination host. It may require the development of ALGs for the interworking with protocols and applications that are not NAT friendly.

Scalability: Only a single IP address is required because all the supported services and connections are multiplexed through it.

Deployment: It does not introduce any changes in the current network infrastructure. The limited resource allocation required could bring economical benefits to operators.

Security: Although private hosts are not exposed to attacks originating in the Internet due to connectivity limitations, a firewall would increase the security of the system.

Disadvantages of the model

Connectivity: A domain resolution operation is required on the remote host to allocate a public IP address from the pool. Moreover, it is very limited to those specific protocols that carry domain information in the user data. The most common are HTTP, SIP, RSTP and SDP. This model may require the addition of proxy servers to process encrypted payload or secure protocols so that the chain of trust is not broken.

Figure 6.2 represents a scenario where multiple public hosts E1-E6 could establish several connections with private hosts H1-H6 using the same public IP address. The services available following this method are the ones listed under the connectivity scope. Examining these protocols in detail it is possible to find domain information inside the payload that can be used for packet forwarding.

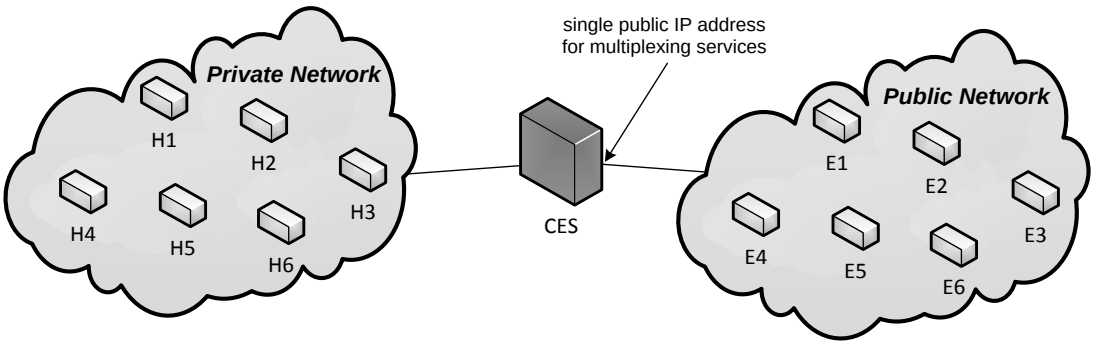


FIGURE 6.2 DOMAIN BASED PACKET FORWARDING

6.4 SRV DNS Query

This model uses a particular type of DNS queries to identify the service requested in the destination host. This type of query is called *Service Record* (SRV) and its main purpose is to provide location of services based on the IP address and the port number.

The SRV record structure and an example are represented below.

```
_service._proto.name TTL class SRV priority weight port target
_sip._udp.example.com. 1000 IN SRV 0 5 5060 sipserver.foo.com.
```

The description of the record fields is the following.

- service: The name of the queried service.
- proto: The transport protocol of the queried service, traditionally TCP or UDP.
- name: The domain name of where the record is valid.
- TTL: The time to live of the record.
- class: The class of the record, it is always IN.
- priority: The priority of the record, the lower value means more preferred.
- weight: A relative weight for records with the same priority.
- port: The port number where the service is located.
- target: The canonical name of the machine that provides the service.

The following lines offer a brief explanation of the advantages and the disadvantages of the model attending the proposed requirements.

Advantages

Flexibility: New protocols can be supported but the compatibility of some applications is still dependent of the implementation of ALGs.

Scalability: A limited consumption of public IP addresses may contribute to solve the address exhaustion.

Deployment: Does not require changes in the current network infrastructure. The limited resource allocation required may bring economical benefits to operators.

Disadvantages

Connectivity: Although the reachability problem appears to be solved there is very limited support for SRV queries in the current applications.

Security: An attacker is able to successfully hijack the state created for another host. The security could be increased by allocating random ports for the requested services.

Despite being standardized only a few applications support and use SRV records. If only it would be compatible with more applications and enjoyed a higher penetration we could have considered creating a model based on SRV queries.

In comparison with the Circular Pool model, both approaches seem to operate in a similar way. However, attending to the scalability and complexity regarding packet forwarding, the SRV model appears as slightly more complex.

7. Circular Pool of Addresses

This chapter focuses on the design of the model called *Circular Pool of Addresses*. The chapter first describes the operation mode and illustrates how the traffic traverses the CES device. Then, a theoretical model is proposed for determining the security of the design. Finally, an analysis in terms of efficiency and scalability is introduced to determine whether the solution attends the design objectives.

7.1 Operation

This section covers the operation mode and explains the policy applied for the allocation of addresses for both incoming and outgoing traffic. The design is highly dependent on the domain name resolutions, from now denoted as a DNS queries.

For outgoing connections, the system allows a private host to establish a communication regardless of a DNS query. In this sense, we can state that the CES device acts as a NAT with multiple public IP addresses. In addition, we could establish different policies to manage the outbound address allocation.

Arbitrary pooling behavior: The public IP address is chosen randomly from the pool of addresses per new connection. The hosts share the whole pool of addresses. If the mapping already exists, an additional port translation operation is performed.
Fixed pooling behavior: Each host is assigned a public IP address that is always used for every outgoing connection. The hosts can share an address. If the mapping already exists, an additional port translation operation is performed.

The proposed mechanisms ensure that outgoing connections will not overlap with each other and the forwarding table stores valid information. As a result, the responses are allowed to traverse the PRGW and properly delivered to the recipient.

Regarding incoming connections, state information is stored each time a DNS query is received. The state comprises different fields such as the originator's IP, the allocated public IP address, the private IP address of the host, status of the entry and a timeout. The public IP addresses are given following a simple circular mechanism,

starting from the beginning of the pool, selecting the next one each time and coming back to the first one upon reaching the end of the pool. When a DNS query is received, the next available address is chosen and marked as in "waiting" status. Because the originator is unknown, the tuple of state information is created with the next information: (unknown, public_IP_in_CES, private_host_IP, waiting, timeout). The actual state information stored in the forwarding table for an active connection includes additional fields regarding the port numbers and the protocol in use. For a better understanding we have decided to use the previously defined tuple of information throughout the rest of the figures.

It is also noteworthy that the *TTL value of the DNS response is set to 0*. The main purpose is to avoid caching in the DNS servers and the remote hosts therefore generating new DNS queries for new connections.

It is very important to understand that an incoming packet that does not match any active connection in the forwarding table but which destination IP address satisfies a matching *waiting* state triggers the creation of new state information and the forwarding of the packet towards the private host. The public IP address is returned to the circular pool for future allocation.

Only the addresses that are not marked in *waiting* state can be reused for new connections. If there are no addresses available by the time a DNS query is received, the connection cannot be established and the circular pool reaches a blocking state. This state is temporary as long as no addresses are available and *only affects the new incoming connections.* The ongoing connections already established in the CES device are not altered and continue to flow without interruptions. As a consequence, the size of the address pool becomes quite important as a limiting factor to avoid entering the blocking state.

In addition, it is possible to establish different policies to determine if a new address should be allocated for a particular host attending to different factors such as:

Global load: This parameter measures the load of the circular pool in terms of connections in waiting status for the whole pool.
Host load: This parameter measures the load of the circular pool in terms of connections in waiting status for a particular host.

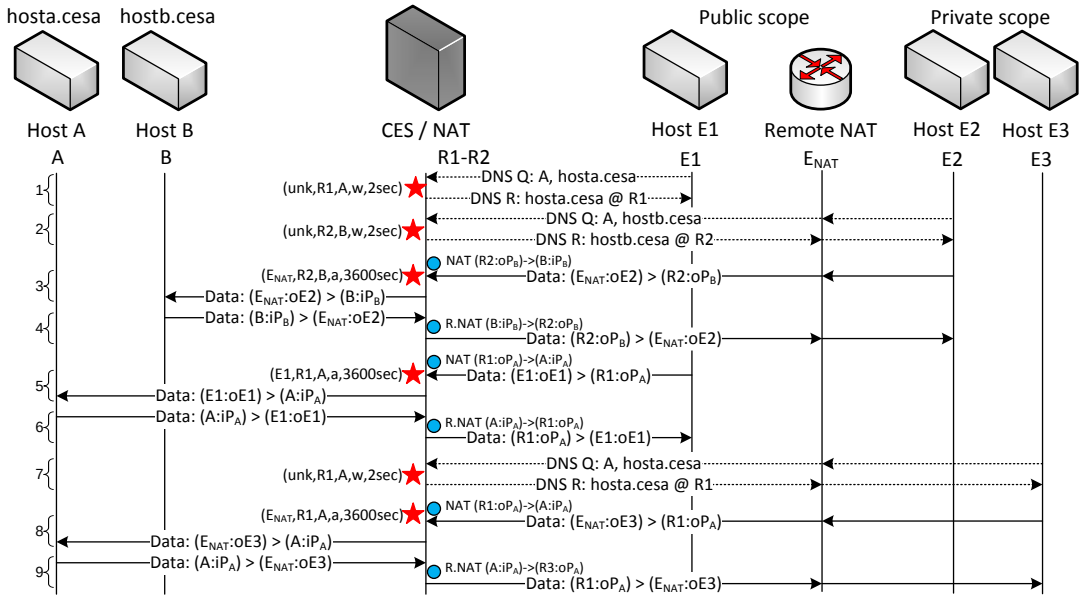
Considering these factors we can determine if a new connection can or cannot be established prior to allocating an IP address from the circular pool. In case of a negative result we can either choose to send a DNS response with an error code or not to send anything at all which will cause the originator to retransmit the original query. The error codes available are the following:

- *2 – Server failure*: The name server was unable to process this query due to a problem with the name server.
- *3 – Name Error*: Meaningful only for responses from an authoritative name server, this code signifies that the domain name referenced in the query does not exist.
- *5 – Refused*: The name server refuses to perform the specified operation for policy reasons. For example, a name server may not wish to provide the information to the particular requester, or a name server may not wish to perform a particular operation (e.g., zone transfer) for particular data.

7.2 Detailed Operation Example

Given the intricacy that might result due to the handling of incoming connections, a highly detailed example is presented to illustrate how our design behaves on scenarios of utter complexity.

Consider the following scenario with three hosts – E1, E2 and E3. The host E1 is directly connected to the Internet and has a public IP address. On the other hand, E2 and E3 are private hosts located in the same network connected to the Internet via the *Remote NAT* and share the same public IP address E_{NAT}. The CES device contains a pool of public addresses R1-R2 as well as two hosts – Host-A and Host-B located in the private network. These hosts can be reached via their FQDN, *hosta.cesa* and *hostb.cesa* respectively. The CES is also the authoritative name-server for the "cesa." zone and therefore handles incoming DNS queries towards this domain. For simplicity any DNS server on the network has been omitted and DNS queries are assumed to be properly forwarded towards the CES. The scenario described is represented in Figure 7.1 that illustrates the timeline for the messages exchanged between all the parties.



A: Private IP of Host-A	iP_A: Local port of Host-A	oP_A: Public port of Host-A
B: Private IP of Host-B	iP_B: Local port of Host-B	oP_B: Public port of Host-B
E1: Public IP of Host-E1	E2: Private IP of Host-E2	E3: Private IP of Host-E3
oE1: Public port of Host-P1	oE2: Public port of Host-E2	oE3: Public port of Host-E3
E_{NAT}: Public IP of E2 and E3	R1-2: Public pool of addresses allocated in CESA	
Tuple state definition (IP source, Public IP in CES, internal host, status(waiting/active), timeout)		

FIGURE 7.1 CIRCULAR POOL - OPERATION

For a more comprehensive understanding consider the sequence of messages taking place right after booting up all devices. There is no previous state information and the forwarding tables are empty.

#1 E1 sends a DNS query requesting an IPv4 address for the domain *hosta.cesa*. Allocation policy is successful and CES selects the next IP address from the pool – *R1*. New waiting state information is stored enabling future traffic addressed to R1 that does not match an ongoing connection to be forwarded internally to Host-A. A DNS response is created returning R1 as the IP address.

#2 Likewise case #1, E2 resolves *hostb.cesa* creating waiting state for *R2*.

#3 Aware of the IP address for Host-B, E2 sends the first data packet to R2. The *Remote NAT* adapts the public IP address to E_{NAT} and the packet is sent to the CES. The packet does not match any ongoing connection in the forwarding table. On the other hand, the destination IP address of the packet matches a *waiting* state in the circular pool. The entry is then marked as *active* and added to the forwarding table. A public-to-private translation for IP address and port

takes place and the mapping is (R2:oP_B) > (B:oP_B). The packet is then forwarded to Host-B. The address R2 is returned to the circular pool.

#4 Host-B sends a response and forwards it to CES that performs a private-to-public translation. The mapping is (B:oP_B) > (R2:oP_B). The packet is forwarded to E_{NAT} and delivered to E2.

#5 Likewise case #3, taking place within the timeout defined in the circular pool, a data packet from E1 addressed to R1 is received in CES. A new entry is added to the forwarding table and the packet is forwarded internally to Host-A. The address R1 is returned to the circular pool.

#6 Likewise case #4, the response generated by Host-A is delivered to E1.

So far we have successfully established two connections with hosts A and B. It is then when Host-E3 attempts to contact Host-A. The messages flow as they follow:

#7 Likewise case #1, E3 attempts to resolve the domain *hosta.cesa*. The query reaches the CES that selects the next IP address from the circular pool – *R1*. New waiting state information is created and a DNS response is sent with the IP address R1. The forwarding table contains now an ongoing connection between E1 and Host-A using the public address *R1* and a waiting state for the same *R1*.

#8 Likewise case #3, the packet originating in E3 traverses the CES device and is forwarded to Host-A. The entry is then marked as *active* and added to the forwarding table. The address R1 is returned to the circular pool.

#9 Likewise case #4, the response from Host-A traverses the CES device and is forwarded to E3.

The example showed above represents the operation mode in an ordered way regardless of external factors affecting the network, such as packet loss or retransmission of packets.

Considering a high load in terms of new incoming connections, it is possible to reach a point where the immediate next address in the pool is still in waiting state. This does not represent an issue to our design because then the next address available will be used instead. Consequently the circular pool can be understood as a centralized system where the resources are extracted when they are allocated and returned for future use when they are released

In addition, it may be possible that some applications fail to operate successfully given the particularities and requirements of the circular pool. In these cases, ALGs must be developed in order to guarantee transparency and compatibility.

7.3 Security Issues and Weaknesses

The proposed Circular Pool model does not need the developing of new technology or protocols to operate successfully. On the contrary, all the protocols and technologies in use are well known, widely deployed and thoroughly tested. This new concept is created upon reusing the current technology in an innovative way.

The heavy reliability on domain resolution may rise some concerns in terms of security and robustness. Despite these facts, our design does not seem to introduce any explicit weakness to the system and yet it is able to handle and neutralize some of the possible attacks. Although it is not presented in any of the figures, an additional layer of security can be deployed, e.g. firewall. A firewall can be installed to neutralize most typical attacks i.e. DoS, DDoS and SYN, protecting at the same time DNS servers from outer networks. In addition, the deployment of a firewall may also improve performance in CES by filtering unwanted connections.

Regarding the security in DNS and despite the fact that it is out of the scope, hereafter we present some hints that may help secure a DNS server that is present in the architecture.

Use of DNS forwarders: Offloads the resolution process from the DNS server to the DNS forwarder benefiting from caching. It is also a good practice to configure a private DNS server to use a forwarder for all those domains for which it is not authoritative.

Use caching-only DNS servers: A caching-only DNS server is not authoritative for any DNS domains and its main purpose is to perform recursion or use a forwarder. It can cache large amount of data significantly improving DNS response times.

Use DNS advertisers: A DNS advertiser is a DNS server that resolves queries for domains for which the DNS advertiser is authoritative.

Use DNS resolvers: A DNS resolver is a DNS server that can perform recursion to resolve domain names for which that DNS server is not authoritative.

Protect DNS from cache pollution: Whereas the caching can improve DNS query performance, if the DNS server cache is polluted with bogus information, users could be forwarded to malicious sites instead of the intended ones.

Limited zone transfers: Zone transfers take place between DNS servers to replicate the information from the primary server into the secondary server. Malicious users could attempt to request a zone transfer dumping the entire zone database file. Zone transfers should be disabled or allowed only to specific servers.

Enable DDNS for secure connections only: Dynamic DNS allows a user to update a resource record in the zone file. A secured connection between the client and the DNS server must be established prior to accepting updates of the DNS information. Otherwise, malicious users could introduce bogus information in the zone file.

Use firewalls to control DNS access: A firewall can be configured to limit the access to the DNS server to a set of matching rules.

Regarding our design, we detected certain vulnerabilities intrinsically related with the operation of the Circular Pool. We considered that they should be at least presented in a way that eases future research to be continued on the topic despite not being the main focus of this research. Based on the operation depicted in Figure 7.1 we identified four different types of attacks as they follow:

Attack #1

Figure 7.2 illustrates how *Attacker-E2* is continuously sending data to a single public IP address - *R1*. The CES does not contain a matching rule for the incoming traffic and drops the packets. When *Host-E1* issues a DNS query for the domain *hosta.cesa*, CES creates state binding the public address R1 with the private address of Host-A. Meanwhile, Attacker-E2 continues sending malicious packets and takes over the connection *reserved* for Host-E1.

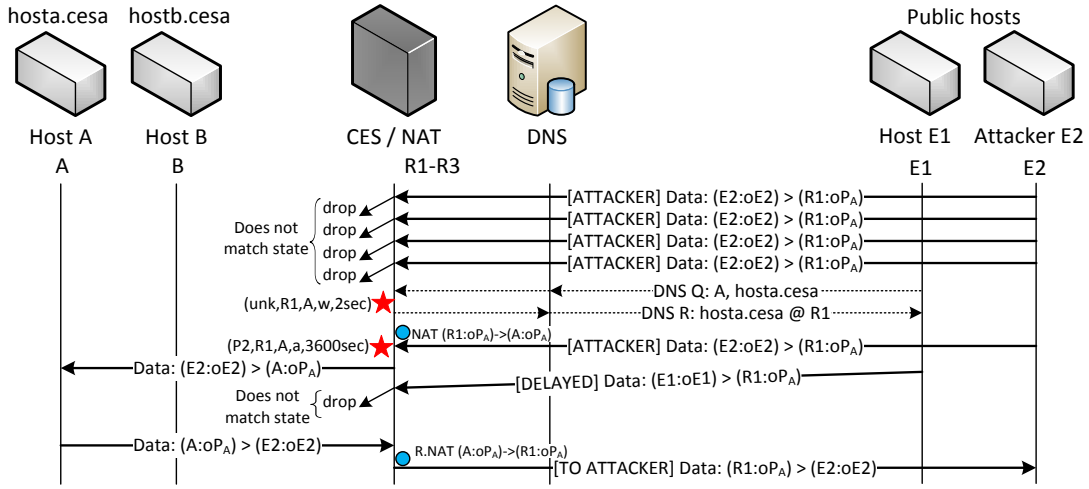


FIGURE 7.2 CIRCULAR POOL – ATTACK #1

Damage: Attacker-E2 hijacks the legitimate connection created by Host-E1 and traverses the CES reaching Host-A. This situation results in DoS from Host-E1 perspective since its packet are dropped by CES because they do not match any state. The security of the server is not *compromised* by this fact.

Vulnerabilities: Only connections in waiting state are vulnerable to this type of attack. Ongoing connections are never affected. A DDoS attack may target a wide range of public IPs from different botnet machines and take over all the reserved states.

Counter-measures and prevention: Detect a non-legitimate source whose packets are repeatedly dropped by the CES. Create an algorithm that generates a blacklist of users based on malicious packet arrival for a time "*T*". During the attack time, discard malicious packets and report the malicious host to a *Trust Management System* [35].

Special cases: Despite the counter-measures it is still possible that the first packets of an attacker are not detected as an attack and treated as legitimate traffic.

Attack #2

Figure 7.3 illustrates how Attacker-E2 is continuously sending DNS queries to different domain names behind the CES that accordingly reserves IP addresses from the pool. Eventually, the pool is depleted because all the public IP addresses are allocated for incoming connections that never occur. As a result, CES is unable to accept new incoming connections since there are not addresses available.

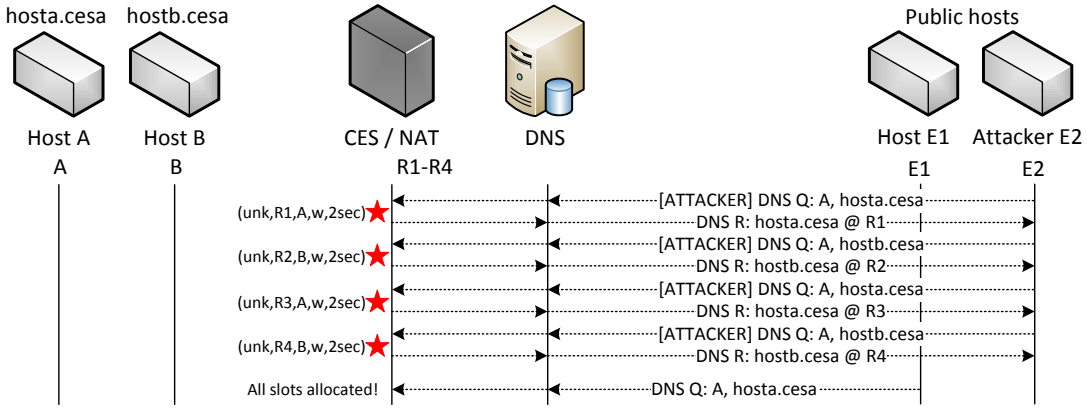


FIGURE 7.3 CIRCULAR POOL – ATTACK #2

Damage: CES device is unable to accept new incoming connections.

Vulnerabilities: Only connections in waiting state are vulnerable to this type of attack, ongoing connections are never affected. An attacker must know at least as many domain names as there are IP addresses in the pool, if only one connection per domain is allowed at a time.

Counter-measures and prevention: Allow a limited number of connections in waiting state per source of the DNS query. Any request that cannot be served should be dropped.

Special cases: If a service is greatly demanded or very popular, it might not be suitable to be located behind a circular pool. The service could suffer some limitations in terms of scalability considering a high rate of new flow arrivals.

Attack #3

Figure 7.4 illustrates how an Attacker-E2 is continuously sending DNS queries via different DNS servers to the same domain name behind the CES. Similarly to Attack #2, the address pool gets depleted and as a result the CES is unable to accept new incoming connections.

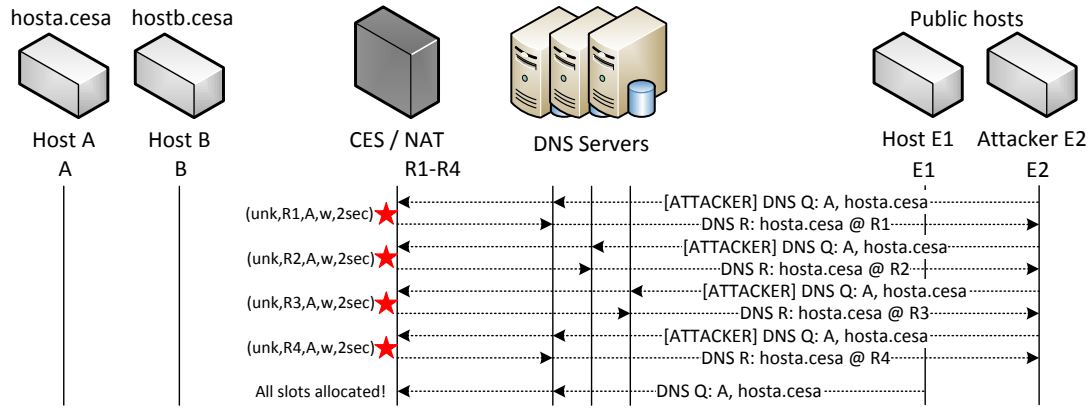


FIGURE 7.4 CIRCULAR POOL – ATTACK #3

Damage: CES device is unable to accept new incoming connections.

Vulnerabilities: Only connections in waiting state are vulnerable to this type of attack, ongoing connections are never affected. An attacker only needs to know a single domain name located behind the CES.

Counter-measures and prevention: Allow a limited number of connections in waiting state per domain. Every request that cannot be served should be dropped.

Special cases: If a service is greatly demanded or very popular, it might not be suitable to be located behind a circular pool.

Attack #4 (3+2)

Figure 7.5 illustrates a combination of Attack #2 and Attack #3. Under these circumstances, an Attacker-E2 is continuously sending DNS requests using different DNS servers querying different domains behind the CES. Similarly to previous attacks, the address pool gets depleted and the CES is unable to accept new incoming connections.

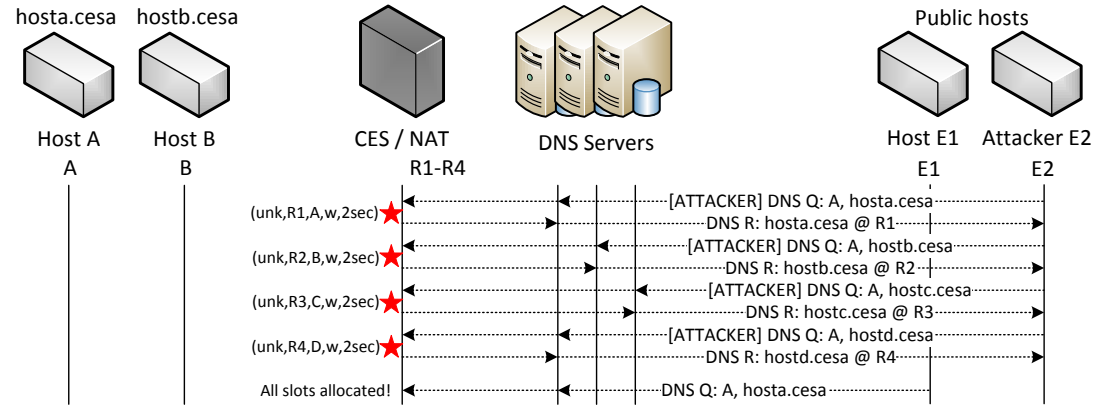


FIGURE 7.5 CIRCULAR POOL – ATTACK #4 (3+2)

Damage: CES device is unable to accept new incoming connections.

Vulnerabilities: Only connections in waiting state are vulnerable to this type of attack, ongoing connections are never affected. An attacker must know at least as many DNS servers and domain names as there are IP addresses in the pool, for the case when only one connection per domain is allowed at a time.

Counter-measures and prevention: Allow a limited number of connections in waiting state per source of the DNS query and domain queried. Any request that cannot be served should be dropped.

Special cases: If a service is greatly demanded or very popular, it might not be suitable to be located behind a circular pool.

Summary of the attacks

As it has been explained, the system could have vulnerabilities when exploiting certain DNS attacks. In order to enhance the protection of the system, local logging and blacklists of malicious hosts can be implemented to deny or filter these connections. With regard to outgoing connections, different policies for public address allocation can be established to make it difficult for potential attackers to obtain an accurate view of the architecture.

In addition, attending to the principle of Trust, the misbehavior of particular users could be reported to either the DNS server managers or even the ISPs. A centralized Trust Management System [35] could receive these reports and apply sophisticated heuristic methods to detect possible DDoS attacks or even create maps of botnets.

7.4 Efficiency and Scalability

Considering efficiency as the property to produce a high ratio of output to input, we can then measure the efficiency of our system in terms of connections per second per IP addresses allocated.

The design of the circular pool allows an incoming connection to be successfully established as long as there are available addresses in the pool. Accordingly the address depletion of the circular pool causes the system to block new incoming connections. Ongoing connections are never affected by this limitation.

Assume that an address is returned to the pool when the first data packet arrives, then the IP address can be immediately reused for the next DNS query. The next example shows in a very clear way the limiting factor in this approach.

Figure 7.6 represents a scenario with a DNS server and two hosts – E1 and E2 – directly connected to the Internet. A public pool consisting of a single IP address – R1 – configured in CES as well as two hosts - A and B located in the private network. These hosts are reachable via their FQDN, *hosta.cesa* and *hostb.cesa* respectively.

Host-E1 attempts to connect to Host-A sending a DNS query. The CES reserves the only IP address available in the pool - R1. From this moment onwards until the first

data packet arrives from Host-E1 or the timeout expires releasing the address, CES enters a blocking state where no new incoming connections can be established.

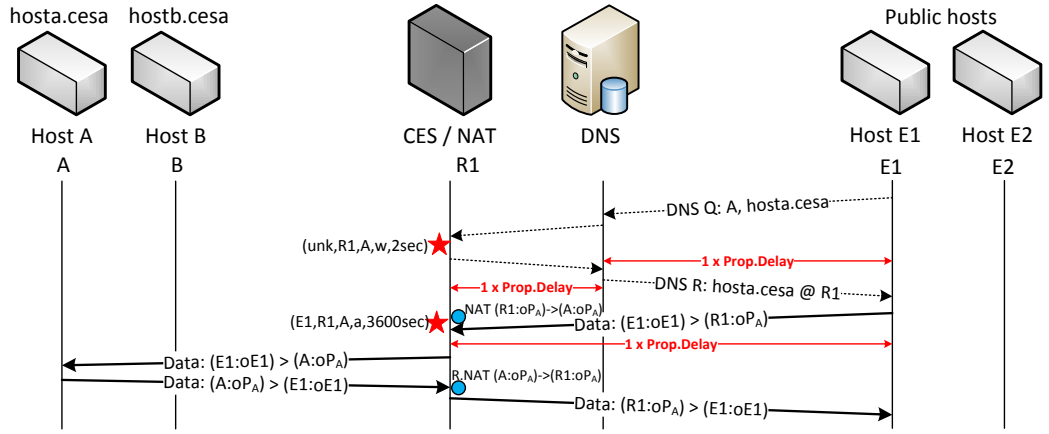


FIGURE 7.6 CIRCULAR POOL – EFFICIENCY

The limiting factor that conditions the efficiency of the system is the time elapsed from the incoming DNS query until the arrival of the first data packet from the user, measured in CES. Figure 7.6 reveals that this *network delay* is the sum of the three propagation delays represented between the CES, the DNS server and the Host-E1.

The *network delay* can also be considered as the *service time* of the circular pool, due to the fact that it measures the time that a resource is not available for use. The following equation represents the upper-bound efficiency of the system:

$$Connections\ per\ second = \frac{Pool\ size}{Service\ time\ (\text{sec})}$$

With the propagation delay of 33 ms and a pool of 1 address as in the previous example, following the efficiency formula the model would be able to process up to:

$$Connections\ per\ second = \frac{1}{(0.033 * 3)\ sec} \approx 10 \text{ new connections per second}$$

For this reason the size of the pool can be proven a crucial factor for avoiding the undesired *blocking state*. The equation indicates a linear increment of the capacity of the system as long as the pool size is enlarged or the network delay is decreased.

8. Evaluation

This chapter focuses on analyzing the implementation of the CES prototype with the legacy interworking using the solution of the Circular Pool. At first we describe the results obtained with the different network protocols and applications. Then, we introduce the additional operations designed in order to grant connectivity to other protocols. After that, a brief summary of the performance analysis is presented to illustrate the scalability of the Circular Pool. Next, some important modifications of the code are introduced. Finally, we present a summary of the testing, evaluation of the requirements and the design objectives.

8.1 Testing the new CES prototype

The following subsections focus on the evaluation of the implemented prototype. The testing is divided into network protocols and applications.

In terms of outgoing connections, the behavior resembles a traditional NAT device connected to the Internet. Hosts connected to the private network share an Internet connection that may be configured with one or more public IP addresses.

Because the circular pool also operates at a higher level than CES, it requires additional state information from both the network and transport protocols. The state information stored in the forwarding table is the following:

- Local IP: The IP address of the local host in the private network.
- Local Port: The port number of the local host in the private network.
- Outbound IP: The IP address of the local host in the public network.
- Outbound Port: The port number of the local host in the public network.
- Remote IP: The IP address of the remote host in the public network.
- Remote Port: The port number of the remote host in the public network.
- Protocol: The type of transport protocol; either TCP, UDP or ICMP.
- Status: Indicates the status of a connection. *Waiting* means an allocated slot in the circular pool, whereas *Active* indicates that packets have flowed both

ways. Set to *Incoming* or *Outgoing* indicates that all traffic so far is unidirectional.

- Timeout: Indicates the time to live in seconds for an entry.
- Timestamp: Contains the last time when the entry was used. Together with the timeout field determines if an entry has expired.
- QoS: Experimental field that can be used for tagging or modifying the Differentiated Services Code Point (DSCP) for realtime multimedia packets.

The scenario submitted to test is represented in Figure 8.1. It comprises two hosts located in different networks and a NAT device that needs to be traversed to enable connectivity. The scenario consists of the following elements:

- hosta: Host located in the private network with IP address 10.10.0.101 and FQDN *hosta.cesa*. The public address assigned for outgoing connections is 1.1.1.11.
- CES/NAT: Serves as a gateway connecting the private and the public network. Contains a pool of public IP address from 1.1.1.11 to 1.1.1.13 allocated for the circular pool. It is also the authoritative name server for the zone "*.cesa"*.
- public: Host located in the public network with IP address 89.141.98.169 and FQDN *jlsantos.no-ip.info*.

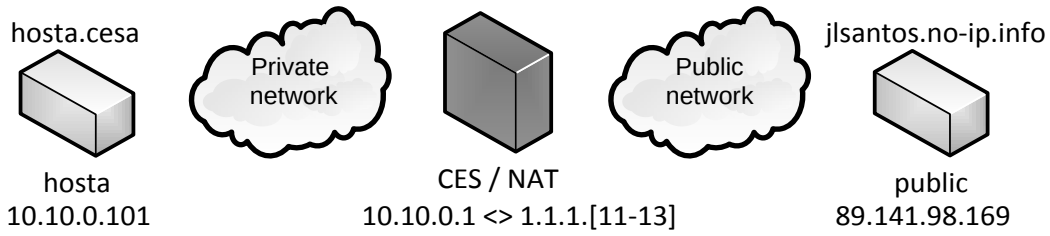


FIGURE 8.1 TESTING SCENARIO

In this section the protocols TCP, UDP and ICMP are submitted to test. There are common valuable parameters that are retrieved from each packet such as the IP source and destination addresses. Attending to the protocol, the additional information required for looking up an entry in the forwarding table is listed as follows:

- TCP: Source and destination port as well as segment flags provide valuable information for the forwarding table. Port numbers identify univocally a

socket connecting two peers. Flag field analysis allows modifications on the entry timeout so the forwarding table is always up to date.

- UDP: Only source and destination port provide valuable information for the forwarding table. Because UDP does not contain any flag signaling the status of the connection, the timeout defined for these entries is always fixed.
- ICMP: Different to TCP and UDP, these packets do not have a port field. Instead, type and code fields can be used for generating valuable information to the forwarding table.

8.1.1 Testing with Network Protocols

In order to test the basic functionality and correct operation of our prototype we will use the application *Netcat - The TCP/IP Swiss Army Knife*. This application allows us to initiate a server/client instance on a given port for both TCP and UDP protocols. With regard to ICMP, the prototype will be tested with the *ping* application. The operation consists of sending an *ICMP echo request* message and receiving an *ICMP echo response* as a response.

These tests are thoroughly explained in Appendix B where we illustrate the command line output of both private and remote hosts as well as the forwarding table of the CES for each of the tests conducted.

As a consequence, here it is only worth mentioning that the results of these tests were mostly successful, with the exception of some ICMP connectivity issues. The prototype revealed that it was able to adequately forward data packets independently of the protocol used, both ways, in conjunction with the Circular Pool.

8.1.2 Testing the Pooling Operation

The following scenario tests the pooling functionality for incoming connections and address allocation per DNS query. From the public network, we will attempt to resolve the domain *hosta.cesa* several times and analyze these responses. The following lines display the console information on the *public* host and the forwarding table in CES.

Remote Host:

```
tester@public:~$ ping hosta.cesa -c 1
PING hosta.cesa (1.1.1.11) 56(84) bytes of data.
64 bytes from 1.1.1.11: icmp_seq=1 ttl=63 time=80.3 ms

--- hosta.cesa ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 80.386/80.386/80.386/0.000 ms

tester@cesvm103:~$ ping hosta.cesa -c 1
PING hosta.cesa (1.1.1.12) 56(84) bytes of data.
64 bytes from 1.1.1.12: icmp_seq=1 ttl=63 time=8.73 ms

--- hosta.cesa ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 8.732/8.732/8.732/0.000 ms

tester@public:~$ dig hosta.cesa
; <<>> DiG 9.7.0-P1 <<>> hosta.cesa
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 51488
..
;; ANSWER SECTION:
hosta.cesa.         0       IN      A       1.1.1.13

tester@public:~$ dig hosta.cesa
; <<>> DiG 9.7.0-P1 <<>> hosta.cesa
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 48818
..
;; ANSWER SECTION:
hosta.cesa.         0       IN      A       1.1.1.11
```

CES:

TABLE 8.1 – CIRCULAR POOL TESTING

NAT TABLE LEGACY STATUS								
LOCAL		OUTBOUND		REMOTE		Prot.	Tout.	Status
IP	Port	IP	Port	IP	Port			
10.10.0.101	20486	1.1.1.11	20486	89.141.98.169	20486	ICMP	60	A
10.10.0.101	20472	1.1.1.12	20472	89.141.98.169	20472	ICMP	60	A
CIRCULAR POOL STATUS								
Local IP		Outbound IP		Timestamp		Timeout		Status
10.10.0.101		1.1.1.13		1326819983.2		2		W
10.10.0.101		1.1.1.11		1326819984.5		2		W

Additional notes: Based on the output of the terminal and the forwarding table, the operation is successful. The entries represented in the table as *active* belong to the ICMP echo requests. In addition, there is another set of entries in *waiting* status that belong to the soft-state created by the circular pool for allocating subsequent incoming connections. The first two ICMP requests allocate the addresses 1.1.1.11 and 1.1.1.12. The dig operation resolves a domain, without sending any data, retrieves the next available address from the pool, in this case 1.1.1.13. Because the pool is

defined with the addresses 1.1.1.11 to 1.1.1.13, the next dig operation retrieves the first available address, in this case 1.1.1.11. According to the design of the circular pool, the results produced by this test are as they were expected.

8.1.3 Testing with Applications

So far we have successfully tested basic TCP, UDP and ICMP operations. The following tests will make use of common applications in order to evaluate the behavior of our prototype.

SSH

The first batch of tests will attempt to establish two simultaneous SSH connections between the devices *hosta* and *public*. The SSH connections are carried on TCP segments and by default on port 22. The following lines display the console information on the hosts and the forwarding table in CES.

Private Host:

```
tester@hosta:~$ ssh jlsantos.no-ip.info
tester@jlsantos.no-ip.info's password:
Linux public 2.6.32-33-generic-pae #71-Ubuntu SMP Wed Jul 20
18:46:41 UTC 2011 i686 GNU/Linux
Ubuntu 10.04.3 LTS

Welcome to Ubuntu!
 * Documentation:  https://help.ubuntu.com/

109 packages can be updated.
81 updates are security updates.

Last login: Tue Jan 17 14:05:14 2012 from 192.168.10.1
tester@cesvm103:~$ who
tester   pts/0        2012-01-17 17:43 (1.1.1.11)
tester   pts/1        2012-01-17 14:05 (192.168.10.1)
tester@public:~$
```

Remote Host:

```
tester@public:~$ ssh hosta.cesa
tester@hosta.cesa's password:
Linux hosta 2.6.32-33-generic-pae #71-Ubuntu SMP Wed Jul 20 18:46:41
UTC 2011 i686 GNU/Linux
Ubuntu 10.04.3 LTS

Welcome to Ubuntu!
 * Documentation:  https://help.ubuntu.com/

Last login: Tue Jan 17 14:06:09 2012 from 192.168.10.1
```

```
tester@cesvm101:~$ who
tester   pts/0        2012-01-18 10:22 (192.168.10.1)
tester   pts/1        2012-01-18 10:23 (89.141.98.169)
tester@hosta:~$
```

CES:

TABLE 8.2 – SSH TCP INCOMING & OUTGOING CONNECTION

NAT TABLE LEGACY STATUS								
LOCAL		OUTBOUND		REMOTE		Prot.	Tout.	Status
IP	Port	IP	Port	IP	Port			
10.10.0.101	39038	1.1.1.11	39038	89.141.98.169	22	TCP	1800	A
10.10.0.101	22	1.1.1.11	22	89.141.98.169	39293	TCP	1800	A

Additional notes: As we can observe based on the output produced on the terminals and the forwarding table, both connections were successfully established.

The next test will attempt to synchronize the clock of the computer via Network Time Protocol (NTP). The information provided by NTP is retrieved in Coordinated Universal Time (UTC) format therefore time zones and daylight saving is out of the scope and must be obtained separately. The NTP packet is carried on UDP datagram on port 123. The following lines display the console information on the host and the forwarding table in CES.

Private Host:

```
tester@hosta:~$ ntpdate pool.ntp.org
20 Jan 17:51:16 ntpdate[1575]: step time server 194.100.2.198 offset
4.448609 sec
tester@hosta:~$
```

CES:

TABLE 8.3 – NTP UDP OUTGOING CONNECTION

NAT TABLE LEGACY STATUS								
LOCAL		OUTBOUND		REMOTE		Prot.	Tout.	Status
IP	Port	IP	Port	IP	Port			
10.10.0.101	123	1.1.1.11	123	213.243.157.156	123	UDP	60	A
10.10.0.101	123	1.1.1.11	123	87.108.20.70	123	UDP	60	A
10.10.0.101	123	1.1.1.11	123	194.100.2.198	123	UDP	60	A

Additional notes: As we can observe based on the output produced on the terminal and the forwarding table, the operation was successful.

Traceroute

Now we will attempt to perform a trace operation from *hosta* to a given domain, i.e. *google.fi* in order to discover the intermediary routers before reaching the destination. The command *traceroute* enables us to perform the operation by sending UDP probe

packets with TTL 1 and incrementing the TTL subsequently until the destination is reached. From the router perspective, forwarding an IP packet with TTL 1 produces an ICMP error message addressed to the originator with the error code set to Time Exceeded.

Private Host:

```
tester@hosta:~$ traceroute google.fi
traceroute to google.fi (209.85.173.94), 30 hops max, 60 byte
packets
 1  * * *
 2  * * *
 3  * * *
 4  * * *
 5  * * *
```

CES:

TABLE 8.4 – TRACEROUTE UDP OUTGOING CONNECTION

NAT TABLE LEGACY STATUS								
LOCAL		OUTBOUND		REMOTE		Prot.	Tout.	Status
IP	Port	IP	Port	IP	Port			
10.10.0.101	49111	1.1.1.11	49111	209.85.173.94	33464	UDP	60	O
10.10.0.101	41192	1.1.1.11	41192	209.85.173.94	33440	UDP	60	O
10.10.0.101	44576	1.1.1.11	44576	209.85.173.94	33443	UDP	60	O
10.10.0.101	46898	1.1.1.11	46898	209.85.173.94	33441	UDP	60	O
10.10.0.101	34937	1.1.1.11	34937	209.85.173.94	33438	UDP	60	O
10.10.0.101	41159	1.1.1.11	41159	209.85.173.94	33460	UDP	60	O
10.10.0.101	38500	1.1.1.11	38500	209.85.173.94	33466	UDP	60	O
10.10.0.101	54841	1.1.1.11	54841	209.85.173.94	33454	UDP	60	O
10.10.0.101	44046	1.1.1.11	44046	209.85.173.94	33468	UDP	60	O
10.10.0.101	40989	1.1.1.11	40989	209.85.173.94	33462	UDP	60	O
10.10.0.101	38770	1.1.1.11	38770	209.85.173.94	33457	UDP	60	O
10.10.0.101	40132	1.1.1.11	40132	209.85.173.94	33446	UDP	60	O
10.10.0.101	51915	1.1.1.11	51915	209.85.173.94	33442	UDP	60	O
10.10.0.101	43462	1.1.1.11	43462	209.85.173.94	33447	UDP	60	O
10.10.0.101	60096	1.1.1.11	60096	209.85.173.94	33451	UDP	60	O

Additional notes: As we can observe based on the output produced on the terminal and the forwarding table, the operation was not successful at all. The problem lies in the inability of the prototype to process the ICMP error messages, to examine the inner faulty IP datagram and finally forward the packet towards the destination. There are multiple scenarios that generate ICMP error messages. The objective of this test was to demonstrate that ICMP error messages do not work without additional processing.

Skype

The following test will submit to test the Skype application. Skype is a widely deployed application for VoIP communications that allows audio calls as well as videoconference and chat. Skype uses a peer-to-peer network where users can become intermediate nodes and forward packets that belong to other user calls in order to overcome obstacles such as NATs or firewalls. The scenario proposed tests the audio call, the videoconference and the chat capabilities.

These tests are explained in Appendix C where we also present screenshots of the private and the remote host as well as the NAT table status for the CES prototype during the process.

As a summary, the prototype enabled both devices to initiate and successfully establish audio and video calls as well as chat sessions. Although not illustrated in the test, file transfer was positively tested too.

On the same topic, regarding the real-time communications field there is an additional research conducted by Petri Leppäaho [15] in the same Department of Communications and Networking. Leppäaho's thesis focuses on the design of ALGs for SIP and FTP communications with CES scenarios.

8.1.4 Testing with HTTP/HTTPS

The following scenarios will submit to test HTTP [6] and HTTPS [27] protocols in order to guarantee full interoperability with web serviceability. HTTP(S) follows the client-server architecture using TCP on the network layer. The operation mode is based on request-response between the client and server. Whereas HTTP/1.0 uses a separate connection for every request-response transaction, HTTP/1.1 is able to reuse a connection multiple times in order to retrieve several elements from the same server. Therefore the latency experienced by HTTP/1.1 is smaller compared to HTTP/1.0. Uniform Resource Locators (URL) are used to address the content.

HTTPS follows the same request-response operation but in this case an SSL connection is first established between the end parties. SSL is a cryptographic

protocol that ensures communication security and provides integrity. A TLS/SSL connection requires a handshake procedure for establishment. [7][32]

During this first test we will connect from *hosta* to the URL http://www.google.fi/ via HTTP and https://encrypted.google.com/ via HTTPS. The outcome of the operation is represented in Figure 8.2, Figure 8.3 and Table 8.5.

Connecting to HTTP Service

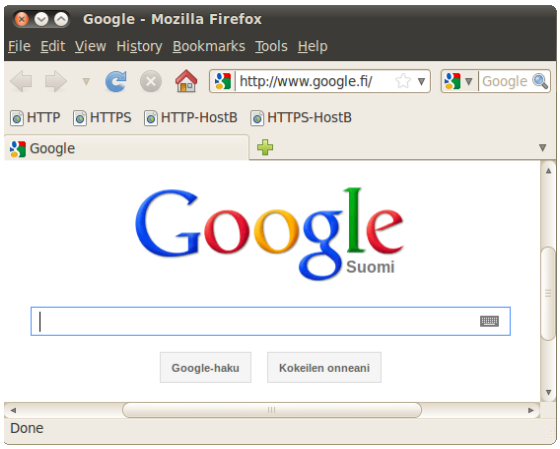


FIGURE 8.2 WEB BROWSER – HOST "HOSTA" AND HTTP TO PUBLIC NETWORK

Connecting to HTTPS Service

FIGURE 8.3 WEB BROWSER – HOST "HOSTA" AND HTTPS TO PUBLIC NETWORK

CES:

TABLE 8.5 – HTTP & HTTPS OUTGOING CONNECTIONS

NAT TABLE LEGACY STATUS								
LOCAL		OUTBOUND		REMOTE		Prot.	Tout.	Status
IP	Port	IP	Port	IP	Port			
10.10.0.101	46802	1.1.1.11	46802	209.85.173.94	80	TCP	1800	A
10.10.0.101	35730	1.1.1.11	35730	173.194.32.31	80	TCP	1800	A
10.10.0.101	57904	1.1.1.11	57904	173.194.32.31	443	TCP	1800	A
10.10.0.101	46803	1.1.1.11	46803	209.85.173.94	80	TCP	1800	A
10.10.0.101	46801	1.1.1.11	46801	209.85.173.94	80	TCP	1800	A
10.10.0.101	48508	1.1.1.11	48508	209.85.173.99	80	TCP	1800	A
10.10.0.101	46392	1.1.1.11	46392	173.194.32.12	443	TCP	1800	A
10.10.0.101	46391	1.1.1.11	46391	173.194.32.12	443	TCP	1800	A
10.10.0.101	46389	1.1.1.11	46389	173.194.32.12	443	TCP	1800	A

Additional notes: As we can observe based on the output produced by the host and the forwarding table, the operation was successful. Destination port 80 corresponds to HTTP while the 443 is associated to HTTPS. Despite the fact that we only loaded a single page, we can observe how the client creates multiple subsequent sockets in order to load the whole content from the page that is composed of text, images and buttons.

In the following test we will connect from *public* to the URL http://hosta.cesa:8080/ces.html via HTTP and once the page has loaded will attempt to connect to https://hosta.cesa:8443/ces_secure.html via HTTPS within the same session meaning that the browser is not restarted. The outcome of the operation is represented below in Figure 8.4, Figure 8.5 and Table 8.6.

Connecting to HTTP Service

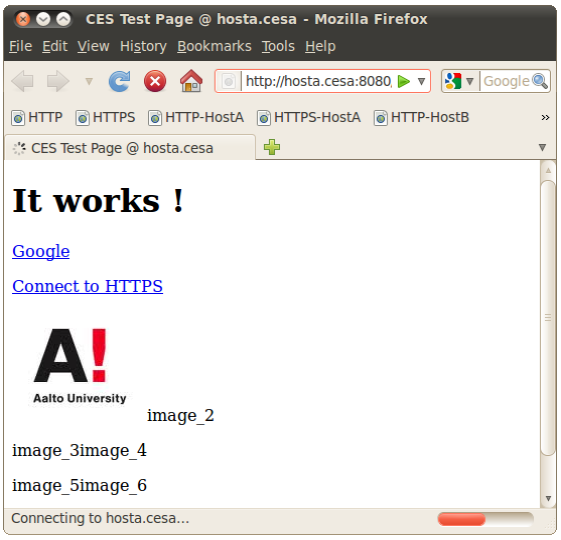


FIGURE 8.4 WEB BROWSER – HOST "PUBLIC" AND HTTP TO "HOSTA"

Connecting to HTTPS Service

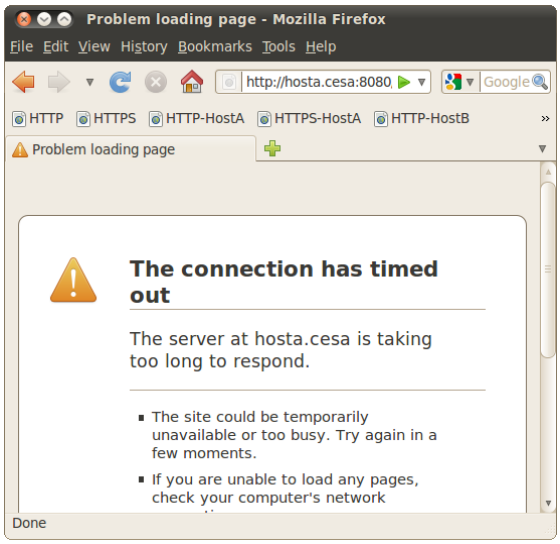


FIGURE 8.5 WEB BROWSER – HOST "PUBLIC" AND HTTPS TO "HOSTA"

CES:

TABLE 8.6 – HTTP & HTTPS INCOMING CONNECTIONS

NAT TABLE LEGACY STATUS								
LOCAL		OUTBOUND		REMOTE		Prot.	Tout.	Status
IP	Port	IP	Port	IP	Port			
10.10.0.101	8080	1.1.1.13	8080	209.85.173.94	53468	TCP	1800	A

Additional notes: The result of the test varies according to the browser used as the testing revealed. In the worst case scenario the browser will not issue a new DNS query because it has been previously *cached*, regardless of the TTL 0 of the response. Under these circumstances the operation simply fails because the incoming connection addressed to the HTTPS server does not match any state and is dropped. In some other cases, different browsers will issue a new DNS query thus allocating a new IP address from the Circular Pool allowing the browser to start a communication even though is prone to fail. On the other hand, and setting this problem aside, we can

verify how the HTML page running on HTTP never finished loading the content while the forwarding table in CES only contains a single entry that belongs to the first packet sent by the browser.

Summary of the HTTP/HTTPS testing

In light of these results and after a thorough analysis we can observe how a browser initiates multiple connections in order to retrieve the page content. In some cases, particularly for incoming connections in CES, this will prevent remote users from getting a smooth and transparent operation. Only the first connection initiated by the browser is granted full connectivity while the new parallel connections will stall. In some other cases the compatibility is even worse considering the user never gets to connect to the site. As a consequence, we can state that the HTTP(S) support is partial or non-supported at all.

Finally, and attending the design objectives some additional mechanism must be implemented in order to guarantee a fully functional and transparent operation allowing HTTP(S) to run smoothly and without further interaction from the user.

8.1.5 Testing with FTP

The following scenarios will submit to test the FTP [21] protocol. FTP is a standard network protocol used for file transferring between two computers. FTP follows client-server architecture and establishes separate control and data connections for communicating. The FTP connections are carried on TCP segments by default on port 21 for control and port 20 for user data. The transport protocol used in this case is TCP. The operational mode can be active or passive.

The next scenario tests the FTP functionality for a client located behind a CES. In this case the FTP server is running in the public network under the domain *jlsantos.no-ip.info*. Considering the two operation modes described before we will first attempt to established and retrieve a file in active and then in passive mode. The outcome of the operation is represented in the following lines.

Active Connection

Private Host:

```
tester@hosta:~$ ftp jlsantos.no-ip.info
Connected to jlsantos.no-ip.info.
220 Welcome to FTP service.
Name (jlsantos.no-ip.info:tester): tester
331 Please specify the password.
Password: *****
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> ls
500 Illegal PORT command.
ftp: bind: Address already in use
```

CES:

TABLE 8.7 – FTP ACTIVE OUTGOING CONNECTION

NAT TABLE LEGACY STATUS								
LOCAL		OUTBOUND		REMOTE		Prot.	Tout.	Status
IP	Port	IP	Port	IP	Port			
10.10.0.101	49701	1.1.1.11	49701	89.141.98.169	21	TCP	1800	A

Additional notes: The operation fails because of the active nature of the connection. The message sent by the client is "*PORT 10.10.0.101,130,53*". Upon receiving this message the server fails to parse it due to the address mismatch between the FTP message and the IP address on the packet. The CES device forwarded the packet and performed a NAT operation on the IP packet without modifying the user data. The outcome of the operation is represented in the following lines.

Passive Connection

Private Host:

```
tester@hosta:~$ ftp jlsantos.no-ip.info
Connected to jlsantos.no-ip.info.
220 Welcome to FTP service.
Name (jlsantos.no-ip.info:tester): tester
331 Please specify the password.
Password: *****
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> passive
Passive mode on.
ftp> get foo
local: foo remote: foo
227 Entering Passive Mode (89,141,98,169,194,208).
150 Opening BINARY mode data connection for foo (49 bytes).
226 Transfer complete.
49 bytes received in 0.01 secs (8.5 kB/s)
```

CES:

TABLE 8.8 – FTP PASSIVE OUTGOING CONNECTION

NAT TABLE LEGACY STATUS								
LOCAL		OUTBOUND		REMOTE		Prot.	Tout.	Status
IP	Port	IP	Port	IP	Port			
10.10.0.101	39502	1.1.1.11	39502	89.141.98.169	21	TCP	1800	A
10.10.0.101	39730	1.1.1.11	39730	89.141.98.169	49872	TCP	1800	A

Additional notes: The operation succeeds because of the passive nature of the connection. The client request changing the operation mode to passive and the server answers with "227 Entering Passive Mode (89.141.98.169,194,208).". Considering that the server is directly connected to the Internet the port is fully reachable and therefore the connection works perfectly because of the public scope of the address given.

The following scenario tests the FTP functionality for a server located behind a CES. In this case the FTP server is installed and running in the private network under the domain hosta.cesa. The client will connect from the public network. Considering the two operation modes described before we will first attempt to establish and retrieve a file in active and then in passive mode. The outcome of the operation is represented in the following lines.

Active Connection

Remote Host:

```
tester@public:~$ ftp hosta.cesa
Connected to hosta.cesa.
220 Welcome to FTP service.
Name (hosta.cesa.ces:tester): tester
331 Please specify the password.
Password: *****
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> get foo
local: foo remote: foo
200 PORT command successful. Consider using PASV.
150 Opening BINARY mode data connection for foo (49 bytes).
226 Transfer complete.
49 bytes received in 0.02 secs (3.0 kB/s)
ftp> ^C
ftp> 221 Goodbye.
```

CES:

TABLE 8.9 – FTP ACTIVE INCOMING CONNECTION

NAT TABLE LEGACY STATUS								
LOCAL		OUTBOUND		REMOTE		Prot.	Tout.	Status
IP	Port	IP	Port	IP	Port			
10.10.0.101	21	1.1.1.11	21	89.141.98.169	56835	TCP	1800	A
10.10.0.101	20	1.1.1.11	20	89.141.98.169	42041	TCP	1800	A

Additional notes: The operation succeeds because of the active nature of the connection. The message sent by the remote client is "*PORT 89.141.98.169,164,57*" therefore the server initiates a new data connection directly to the given IP address and port. In this case the NAT operation does not prevent this scenario from working.

Passive Connection

Remote Host:

```
tester@public:~$ ftp hosta.cesa
Connected to hosta.cesa.
220 Welcome to FTP service.
Name (hosta.cesa.ces:tester): tester
331 Please specify the password.
Password: *****
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> passive
Passive mode on.
ftp> get foo
local: foo remote: foo
227 Entering Passive Mode (10,10,0,101,86,184).
ftp: connect: Connection timed out
```

CES:

TABLE 8.10 – FTP PASSIVE INCOMING CONNECTION

NAT TABLE LEGACY STATUS								
LOCAL		OUTBOUND		REMOTE		Prot.	Tout.	Status
IP	Port	IP	Port	IP	Port			
10.10.0.101	21	1.1.1.11	21	89.141.98.169	36931	TCP	1800	A

Additional notes: Similarly to what happened with the first FTP test case "Active Outgoing Connection" the connection fails because of the NAT. The server responds with the message "*227 Entering Passive Mode (10.10.0.101,86,184).*". The client is unable to establish a connection because of the mismatching in type and nature of the address. The client should start a connection with an address configured within the circular pool to ensure at least the packets are delivered to the CES.

Summary of the FTP testing

After submitting to test the FTP protocol for file transferring we can conclude that the support is only partial in the sense that two out of four tests were successful. The scenarios that did not succeed to enable a smooth FTP transaction were:

- Outgoing active mode: The client is located in the private network and expects the server to send the information while waiting in a listening state.
- Incoming passive mode: The server is located in the private network and expects a client to connect while waiting in a listening state.

The problems originate from the non-existing mappings in the forwarding table preventing incoming traffic from being delivered to the correct IP.

Finally, following the dictate of the design objectives, additional research is required in order to guarantee a smooth FTP operation under all circumstances. [15]

8.2 Application Layer Gateway

In the light of the results obtained in the previous section, we are obliged to create some additional processing that guarantees total compatibility with applications and protocols that fail to operate successfully on standard conditions. These operations are called Application Layer Gateways and their purpose is to enable communication through NATs and firewalls transparently to the end user.

ALGs are triggered by e.g. FTP, SIP or SDP protocols. These protocols convey information related to the local interface where the socket is bound and therefore the gateway must adapt the scope from private to public and vice versa prior to forwarding the packets. In other cases, it is also required to establish additional mappings in the forwarding table to achieve interoperability.

8.2.1 Application Layer Gateway for ICMP

This section explains how incoming ICMP packets are processed in order to overcome the compatibility issues that appeared during the testing. These problems arose when using the *traceroute* application but they are extensible to other scenarios.

The testing revealed that ICMP error packets originated in the public network are never delivered to the private host because they get dropped by the CES. To that end, we developed the following Application Layer Gateway for CES.

The operation is triggered by an incoming ICMP packet that containing a faulty IP datagram inside. The error packet is extracted and a lookup operation with the forwarding table is performed. Upon a matching result, we can conclude that the packet originated from a host behind the CES. The values of the error packet are accordingly updated with the information retrieved from the forwarding table. The external ICMP packet is modified as well. The private host receives the packet and become aware of the faulty operation. On the other hand, if the lookup process does not return a matching result, the packet is dropped by the CES.

Figure 8.6 represents the flow diagram of the ICMP application layer.

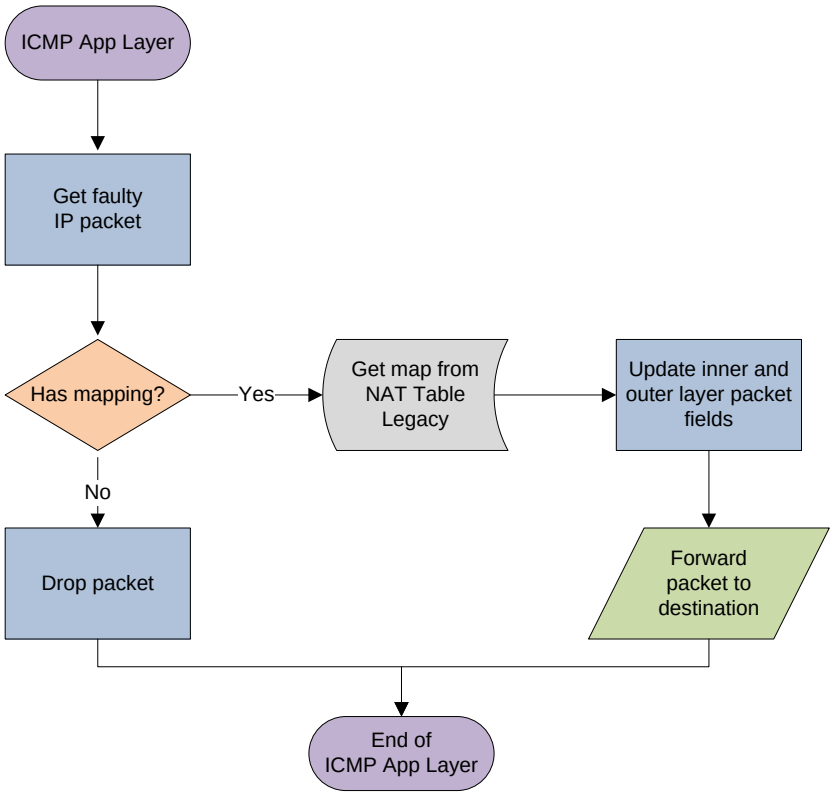


FIGURE 8.6 ALG ICMP - FLOW DIAGRAM

Now we introduce an example of *traceroute* to confirm the correct behavior of the application layer. The following lines display the console information on the host and the forwarding table in CES.

Private Host:

```
tester@hosta:~$ traceroute kosh.aalto.fi -q 1
traceroute to kosh.aalto.fi (130.233.224.196), 30 hops max, 60 byte
packets
 1  cesvm102.local (10.10.0.1)  71.286 ms
 2  89.141.98.169.dyn.user.ono.com (89.141.98.169)  174.646 ms
 3  *
 4  gw-rs.research.netlab.hut.fi (195.148.124.129)  185.961 ms
 5  funet-rtr.research.netlab.hut.fi (195.148.124.6)  195.953 ms
 6  gw-2-10g-funet-main.aalto.fi (130.233.231.190)  191.201 ms
 7  kosh.org.aalto.fi (130.233.224.196)  200.941 ms
```

CES:

TABLE 8.11 – TRACEROUTE UDP OUTGOING CONNECTION WITH ALG ICMP

NAT TABLE LEGACY STATUS								
LOCAL		OUTBOUND		REMOTE		Prot.	Tout.	Status
IP	Port	IP	Port	IP	Port			
10.10.0.101	32946	1.1.1.11	32946	130.233.224.196	33438	UDP	60	O
10.10.0.101	37234	1.1.1.11	37234	130.233.224.196	33449	UDP	60	O
10.10.0.101	47202	1.1.1.11	47202	130.233.224.196	33447	UDP	60	O
10.10.0.101	35597	1.1.1.11	35597	130.233.224.196	33443	UDP	60	O
10.10.0.101	49439	1.1.1.11	49439	130.233.224.196	33444	UDP	60	O
10.10.0.101	56521	1.1.1.11	56521	130.233.224.196	33439	UDP	60	O
10.10.0.101	60439	1.1.1.11	60439	130.233.224.196	33445	UDP	60	O
10.10.0.101	36516	1.1.1.11	36516	130.233.224.196	33446	UDP	60	O
10.10.0.101	51784	1.1.1.11	51784	130.233.224.196	33450	UDP	60	O
10.10.0.101	46978	1.1.1.11	46978	130.233.224.196	33437	UDP	60	O
10.10.0.101	48640	1.1.1.11	48640	130.233.224.196	33436	UDP	60	O
10.10.0.101	56130	1.1.1.11	56130	130.233.224.196	33435	UDP	60	O
10.10.0.101	56813	1.1.1.11	56813	130.233.224.196	33441	UDP	60	O
10.10.0.101	41113	1.1.1.11	41113	130.233.224.196	33440	UDP	60	O
10.10.0.101	56286	1.1.1.11	56286	130.233.224.196	33448	UDP	60	O
10.10.0.101	41576	1.1.1.11	41576	130.233.224.196	33442	UDP	60	O

Additional notes: As we can observe based on the output produced by the terminal the operation was successful. Regarding the forwarding table and despite succeeding the entries are in “Outgoing” status because there was never an incoming UDP datagram but an ICMP packet instead. To that extent we can assure that ICMP error messages are forwarded to the originator in both outgoing and incoming fashion.

8.2.2 Application Layer Gateway for HTTP(S)

In this section, we conduct a deep analysis of the Web protocols in order to understand their behavior and then to develop alternatives methods for overcoming the connectivity issues detected in Section 8.1.4.

At first we developed a new Application Layer Gateway for HTTP by applying basic heuristic algorithms which eventually resulted into a false start. The implemented model and tests case are explained in Appendix D.

As a result we had to focus on finding an alternative solution. Ultimately we borrowed one of the concepts previously researched, the *Domain Based Packet Forwarding* so that the Web traffic could be received on a fixed IP address. In conjunction with a static mapping in the forwarding table, all Web traffic matching with the HTTP/HTTPS ports is statically forwarded to an internal HTTP-Proxy server located in the private network.

An HTTP-Proxy server acts as an intermediary for requests originating in clients attempting to locate and fetch content from other servers. The operation resembles a *man-in-the-middle* scenario. The client contacts the proxy and requests certain content; the proxy first determines the location, then proceeds to retrieve the content and finally forwards it to the client who originated the request. The proxy server may also keep a copy of the content in cache memory which decreases the network traffic, HTTP server load and increases the responsiveness with the client.

There are several types of HTTP-Proxies and each of them has a very particular operation mode. The one that best serves our purpose is called *reverse proxy*.

A reverse proxy is a type of proxy server that retrieves content from one or several servers on behalf of a client and forwards the data to the client as if it would be originated from the proxy itself. Whereas a forwarding proxy performs intermediary functions for certain clientele fetching and caching content from the Internet, a reverse proxy operates as an intermediary for the associated servers returning only the information available in these.

The advantages of having a reverse proxy serving multiple HTTP servers are many:

- Hides the existence of the given HTTP server to the originating client. The client does not have any knowledge of the presence of the HTTP proxy in the network.
- SSL encryption can be offloaded into a different system that is conveniently better suited for these operations, sometimes even equipped with SSL acceleration hardware, enabling *Secure HTTP*.
- Enables load balancing capabilities so that a single request can be split and offloaded to several servers. It is possible that the URL must be modified to address a given resource in the right server.
- Reduces the load of HTTP servers and network traffic by caching the content. The performance and scalability are greatly enhanced by this technique. The proxies are sometimes referred as *web accelerators*.
- Optimizes data transfer by applying compression mechanism reducing the size of the information.
- Because its transparent nature to the user, it is possible to allocate multiple HTTP servers behind a single IP address reusing the same ports. This is extremely beneficial in scenarios where a NAT device is present because clients located in the public network are able to fetch content from servers in a private network without direct connectivity. In this case only a port forwarding operation is required so that HTTP traffic is forwarded internally to the reverse proxy.

Figure 8.7 presents in a very simplistic manner an example of a reverse proxy.

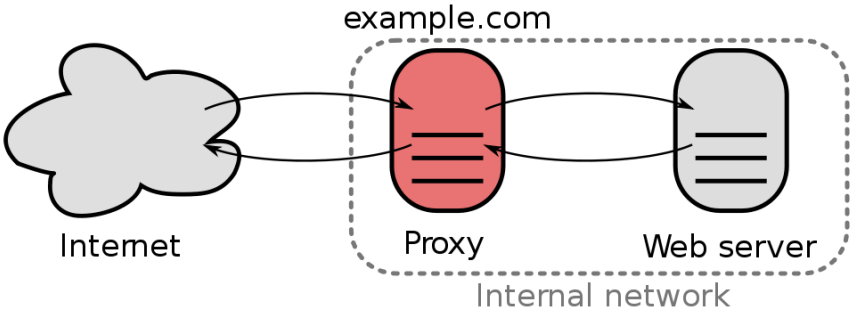


FIGURE 8.7 REVERSE HTTP PROXY ARCHITECTURE

In Figure 8.8 we illustrate the packet flow when *public* host attempts to connect with web services running on both *hosta* and *hostb*. Note that the domain queried in the DNS request has changed to "*hosta.web.cesa*" and "*hostb.web.cesa*" respectively. Special domain names are needed in order to differentiate web traffic from regular traffic, and forward it to the reverse proxy. To that end, the address indicated in the DNS response does not belong to the circular pool but instead is treated separately as a *premium* service with a static IP address and port forwarding enabled. This differentiation enables CES to set a static mapping on HTTP and HTTPS ports on that particular address and forward the traffic directly to the proxy.

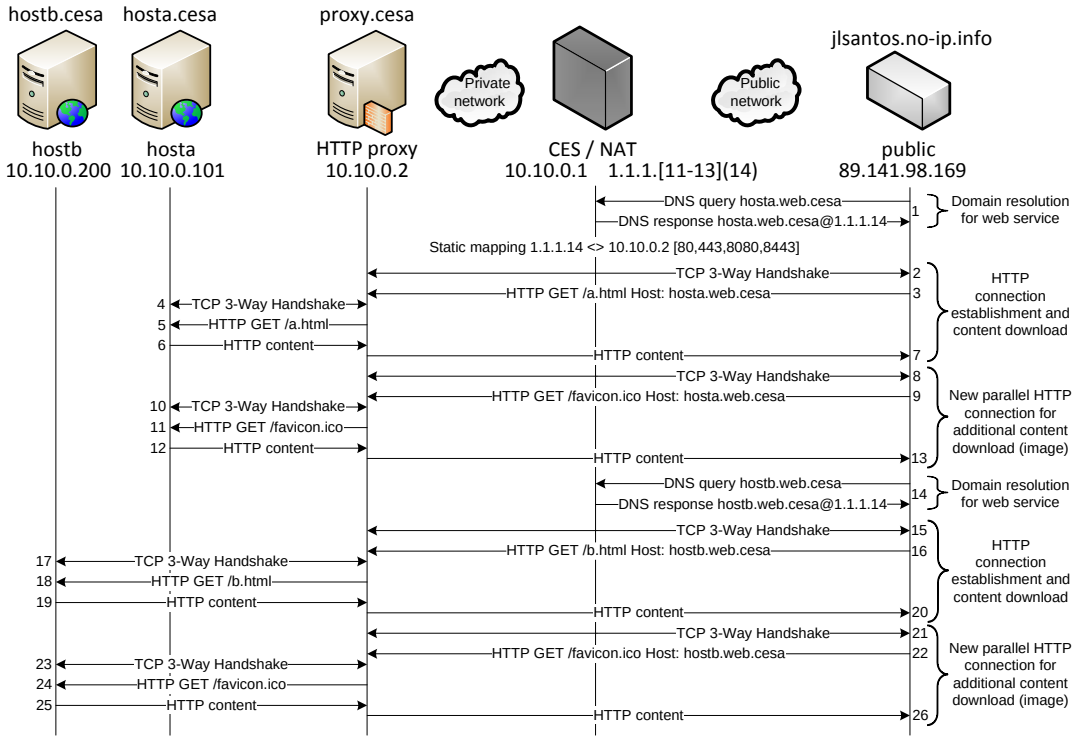


FIGURE 8.8 REVERSE HTTP PROXY OPERATION

#1 Domain resolution request/response for the domain *hosta.web.cesa*

#2 TCP Three-way handshake for onnection establishment.

#3 HTTP "*GET /a.html Host: hosta.web.cesa*" message to download content.

#4 Proxy parses the request and according to its configuration determines that *hosta* in the private network is the intended recipient. Proxy initiates a TCP Three-way handshake for connection establishment.

#5 Proxy forwards the GET request adding extra HTTP headers referring to the original client with the IP address 89.141.98.169.

#6 HTTP server in *hosta* forwards the content to the proxy.

#7 Proxy forwards the content retrieved from *hosta* towards *public*.

#8 - #13 The process repeats itself from previous steps #2 - #7 as the intended server is *hosta*.

#14 - #20 The process repeats itself from previous steps #1 - #7 although in this case the parsing of the GET message casts *hostb* as result. The same methods apply as explained before with *hostb* as the ultimate recipient.

#21 - #26 The process repeats itself from previous steps #15 - #20 as the intended server is *hostb*.

Summarizing the HTTP compatibility we can confirm that introducing the HTTP-Proxy into the current scenario solves all the problems and incompatibilities detected when using Web protocols. In addition, this tool supports very versatile and fine grained configuration that users could benefit from such proxy-cache. Under this mode, the proxy is able to cache Internet content and send a compressed response to the client, reducing also the congestion on public networks.

8.2.3 Application Layer Gateway for FTP

This section explains how the FTP transactions are processed in order to overcome the compatibility issues revealed during the testing process. In Section 8.1.5 two major issues were discovered that prevented the prototype from working adequately. The solution that we propose here established the foundations for enabling FTP transactions across CES-to-CES communications included in Leppäaho's work [15]. The specific details about the FTP operation mode are described below:

In the active mode, the client indicates the server in the control connection where data are expected to be received. The server is the one that actively initiates a connection with the client and sends the data. The client sends the message of the form "*PORT IPaddress,12,34*".

In the passive mode, the client requests from the server the address and port where to download the data from. Opposite to the previous case, the client initiates a new connection with the server in order to download the data. The server responds with a message of the form "*227 Entering Passive Mode (IPaddress,12,34).*".

In both cases, the host always uses the local IP address where the server is bound. Considering the problems only arise when the host is located in a private realm, the problem lies in the need for adapting the private IP address conveyed in the message to the correspondent public IP address in the public realm. In addition, a new mapping has to be created in the forwarding table for the subsequent data connection.

In order to create the extra mapping, the ALG has to determine the port that will be used for the data transaction. Note the port number is 16 bits long. According to the example "*IPaddress,12,34*", the port number can be extracted applying a basic calculation. First we will extract the Most Significant Byte (MSByte) "12" and then the Least Significant Byte (LSByte) "34". The following operation gives us the result.

$$port\ number = (256 * MSByte) + LSByte = (256 * 12) + 34 \rightarrow 3106\ \text{TCP}.$$

Note that TCP uses the fields Sequence (SEQ) and Acknowledgement (ACK) together with the segment length in order to provide ordering of packets and reliability. Because the ALG modifies the payload of the TCP connection, it is possible that the length of the packet has changed. In that case, the fields SEQ and ACK have to be accordingly modified to account for the offset introduced so that the TCP flow remains undisrupted.

The operations performed by the ALG are as follow:

1. The ALG is triggered in the CES by an outgoing packet on the FTP control port with a message starting with either "*PORT*" or "*227 Entering Passive Mode"*.
2. An address translation operation is performed over the IP address conveyed in the payload. The scope is adapted from private to public. An additional mapping is created for the incoming data connection.
3. The ALG calculates the offset introduced in the packet length as follows.

$$\mathbf{Offset}_{\mathbf{PRODUCED}} = \mathbf{Length}_{\mathbf{NEW}} - \mathbf{Length}_{\mathbf{CURRENT}}$$

4. Then the calculated value is added to a table that keeps state information about active FTP transactions. If the connection existed already, then the new offset is calculated based on the sum of the introduced offset plus the previous one.

$$\mathbf{Offset}_{\mathbf{NEW}} = \mathbf{Offset}_{\mathbf{PRODUCED}} + \mathbf{Offset}_{\mathbf{OLD}}$$

5. The fields SEQ and ACK of the *subsequent* control packets need to be modified so that the TCP flow is not disrupted. Attending to the direction of the packet the operation differs:
    - To public realm: **ACK $_{NEW}$ = ACK $_{CURRENT}$ – Offset $_{NEW}$**
    - To private realm: **SEQ $_{NEW}$ = SEQ $_{CURRENT}$ + Offset $_{NEW}$**

Suppose an outgoing FTP active connection originates from the private realm towards the Internet. Then, the client initiates two data transfers such as listing a directory and retrieving a file.

Figure 8.9 illustrates this scenario. Assume "*S*" as the sequence number, "*A*" as the acknowledgement number and "L" as the length of the segment.

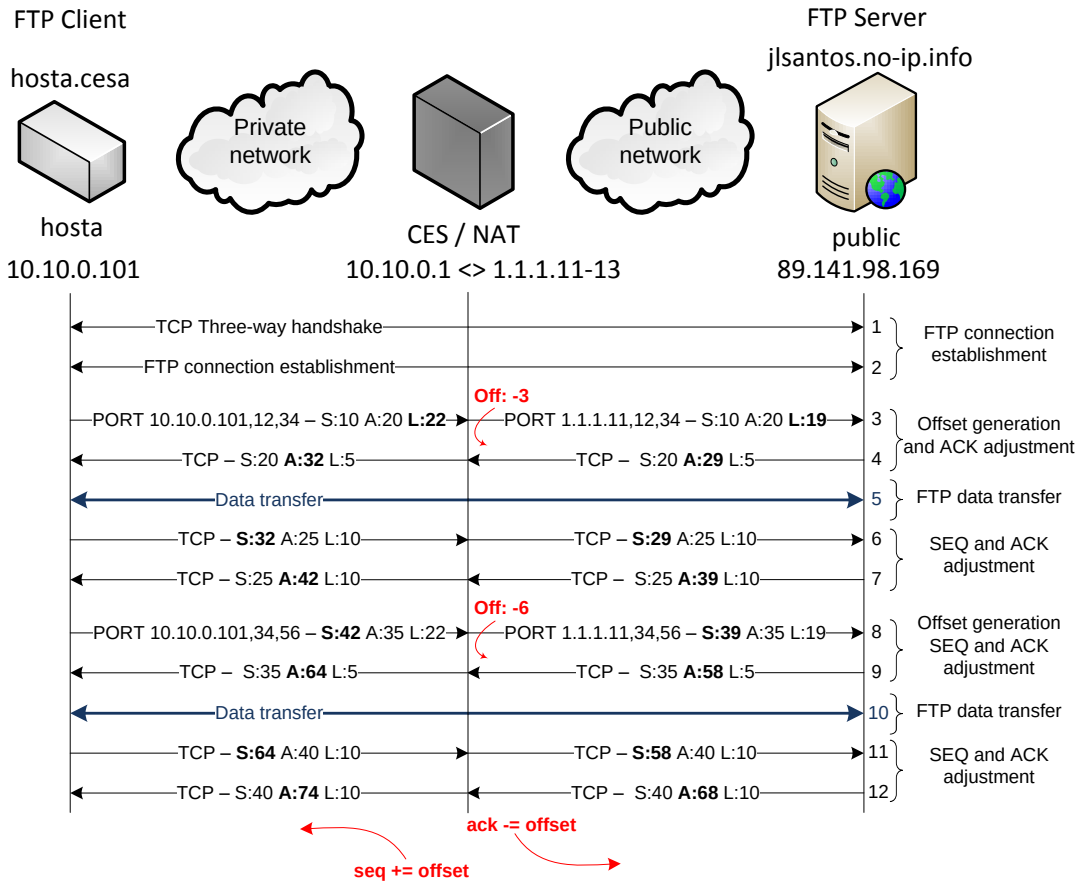


FIGURE 8.9 ALG FTP - PACKET SEQUENCE

#1 Domain resolution request/response for the domain *jlsantos.no-ip.info*, TCP Three-way handshake for connection establishment.

#2 The client authenticates himself in the FTP server and logs in.

#3 The client requests a directory listing on the remote side and a message "*PORT 10.10.0.101,12,34*" is sent indicating the active connection. ALG FTP is triggered in CES, the TCP payload is changed to "*PORT 1.1.1.11,12,34*" resulting an offset of "*-3*" due to the difference in length of the strings. A new mapping is added to the forwarding table to allow an incoming connection on that IP address and port.

#4 The incoming TCP segment that contains an ACK of 29 has to be modified according to the offset; the resulting ACK is 32. The packet is modified and forwarded to the client.

#5 The FTP data connection takes place as a regular outgoing connection.

#6 The outgoing TCP segment that contains an SEQ of 32 has to be modified according to the offset; the resulting SEQ is 29. The packet is modified and forwarded to the server.

#7 Likewise case #4. ACK is updated resulting in 42.

#8 Likewise case #3. The introduced offset is "-3" again. The new offset value is updated to "*-6*". The SEQ is updated with the *old offset* resulting in 39.

#9 Likewise case #4. ACK is updated resulting in 58.

#10 Likewise case #5.

#11 Likewise case #6. SEQ is updated resulting in 58.

#12 Likewise case #4. ACK is updated resulting in 68.

Figure 8.10 represents the flow diagram of the application layer.

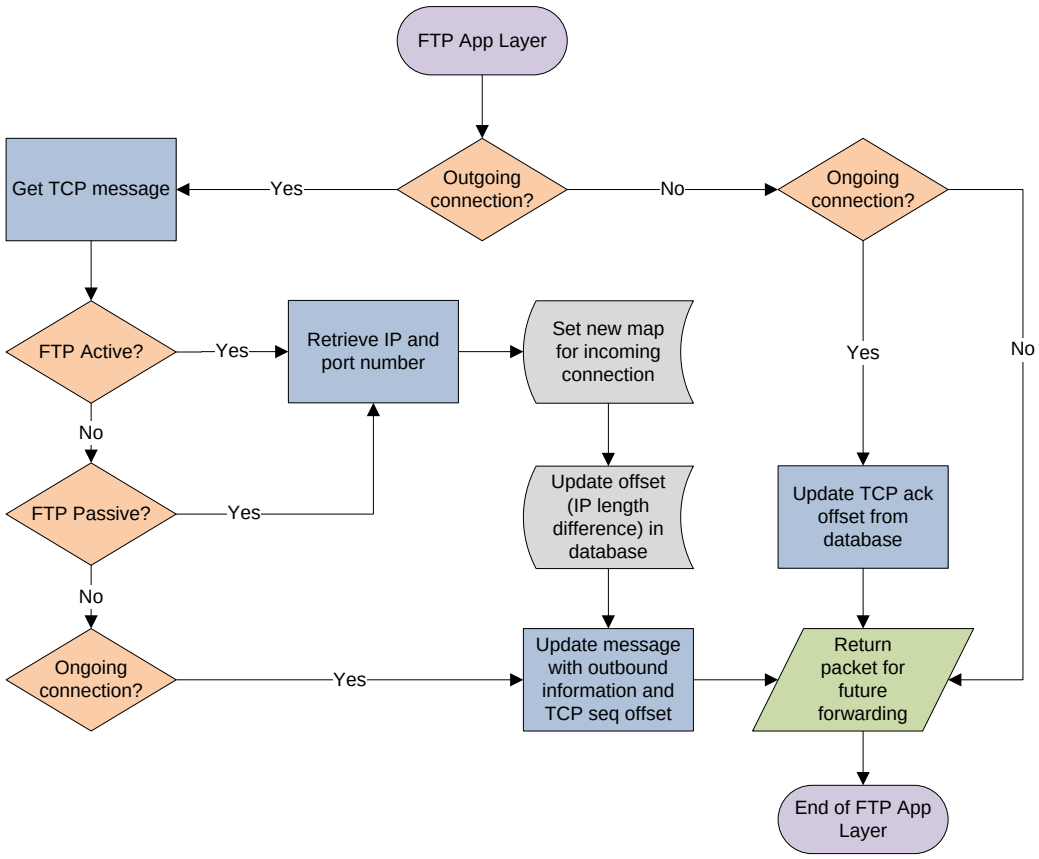


FIGURE 8.10 ALG FTP – FLOW DIAGRAM

Now we introduce an example to confirm the correct behavior of the application layer. The scenario is the same represented in Figure 8.9 where a device located in the private network will attempt to initiate an FTP transaction with a server on the public realm. Both of the operation modes available in FTP will be submitted to test starting with *active* and followed by *passive*. The outcome of the operation is represented in the following lines.

Private Host:

```
tester@hosta:~$ ftp jlsantos.no-ip.info
Connected to jlsantos.no-ip.info.
220 Welcome to FTP service.
Name (jlsantos.no-ip.info:tester): tester
331 Please specify the password.
Password: *****
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> get foo
local: foo remote: foo
200 PORT command successful. Consider using PASV.
150 Opening BINARY mode data connection for foo (49 bytes).
226 Transfer complete.
49 bytes received in 0.01 secs (4.2 kB/s)
```

```
ftp> get foo
local: foo remote: foo
200 PORT command successful. Consider using PASV.
150 Opening BINARY mode data connection for foo (49 bytes).
226 Transfer complete.
49 bytes received in 0.01 secs (8.8 kB/s)
ftp> passive
Passive mode on.
ftp> get foo
local: foo remote: foo
227 Entering Passive Mode (89,141,98,169,197,72).
150 Opening BINARY mode data connection for foo (49 bytes).
226 Transfer complete.
49 bytes received in 0.02 secs (2.6 kB/s)
ftp> get foo
local: foo remote: foo
227 Entering Passive Mode (89,141,98,169,186,2).
150 Opening BINARY mode data connection for foo (49 bytes).
226 Transfer complete.
49 bytes received in 0.01 secs (8.8 kB/s)
ftp> ^C
ftp> 221 Goodbye.
```

CES:

TABLE 8.12 – FTP ACTIVE & PASSIVE OUTGOING CONNECTION WITH ALG FTP

NAT TABLE LEGACY STATUS								
LOCAL		OUTBOUND		REMOTE		Prot.	Tout.	Status
IP	Port	IP	Port	IP	Port			
10.10.0.101	38968	1.1.1.11	38968	89.141.98.169	21	TCP	1800	A
10.10.0.101	50736	1.1.1.11	50736	89.141.98.169	47618	TCP	12	A
10.10.0.101	59720	1.1.1.11	59720	89.141.98.169	50504	TCP	12	A
10.10.0.101	58541	1.1.1.11	58541	89.141.98.169	20	TCP	12	A
10.10.0.101	50074	1.1.1.11	50074	89.141.98.169	20	TCP	12	A
APP LAYER FTP STATUS								
		OUTBOUND		REMOTE		OFFSET		
		IP	Port	IP	Port			
		1.1.1.11	38968	89.141.98.169	21	-6		

Additional notes: The operation succeeds once the application layer has established the proper conditions in the forwarding tables and accordingly modified the user data of the TCP segments. Analyzing the previous table we can observe how the first entry belongs to the FTP control connection, the following two connections are related to the active connection whereas the last ones correspond to the passive connection. Note the remote port in the *public* host is set to 20 as an indicator of data port in FTP.

It is also worth mentioning that the application layer also works whenever the FTP server is located in the private realm and a remote client uses the passive mode to initiate a data connection from the public realm.

8.3 Performance Analysis

This section introduces the performance overview of the implemented solution. As it was described in Chapter 7 – *Circular Pool of Addresses*, the performance of the system is directly bound to two factors, the pool size and the network delay. A detailed explanation of how the tests were conducted can be found on my other research project "Special Assignment – Testing and Measurements of CES Interworking with Legacy Networks" [16] that was specifically designed to cover this topic. Hereafter only the main points are covered in order to introduce some graphical results obtained based on those tests. Approximately 200 tests were conducted to retrieve the data and elaborate the corresponding graphs.

Figure 8.11 illustrates how the network delay can be calculated for mathematical analysis if we neglect the processing time on the represented nodes.

Network Delay: Delay-1 + Delay-2 + Delay-3

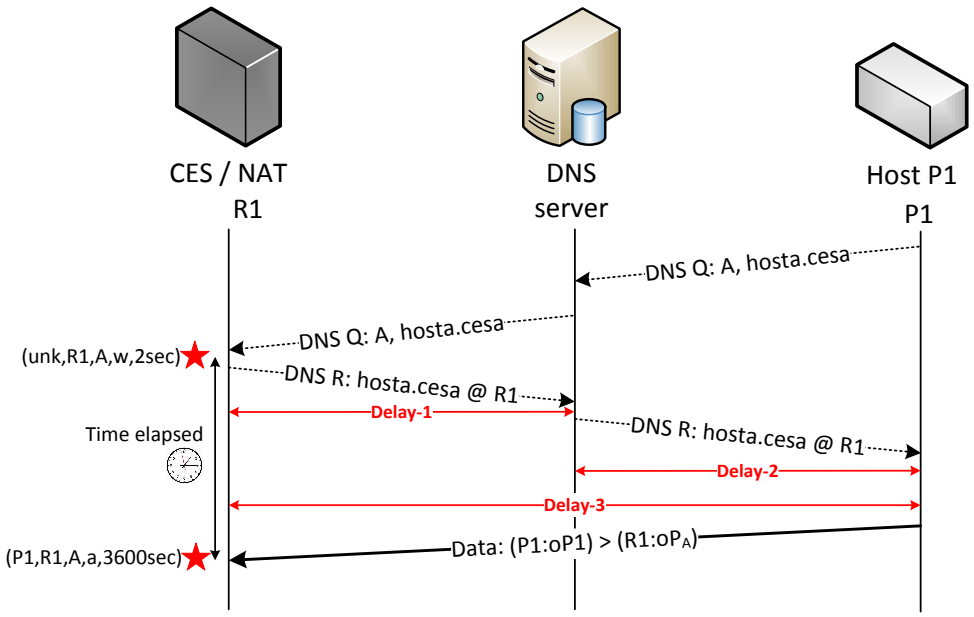


FIGURE 8.11 PERFORMANCE ANALYSIS – NETWORK DELAY

Next sections will submit to analysis how the system responds when the network conditions are modified in terms of pool size, network delay and offered load.

The testing scenario is represented in Figure 8.12 and consists of a client located in the public network attempting to connect to a server behind a CES device. The public

host will issue DNS queries in order to initiate a UDP connection that is echoed back by the server. If the name resolution fails due to unavailability of IP addresses in the Circular Pool, the process is reattempted up to four [1] additional times before the connection is marked as failure.

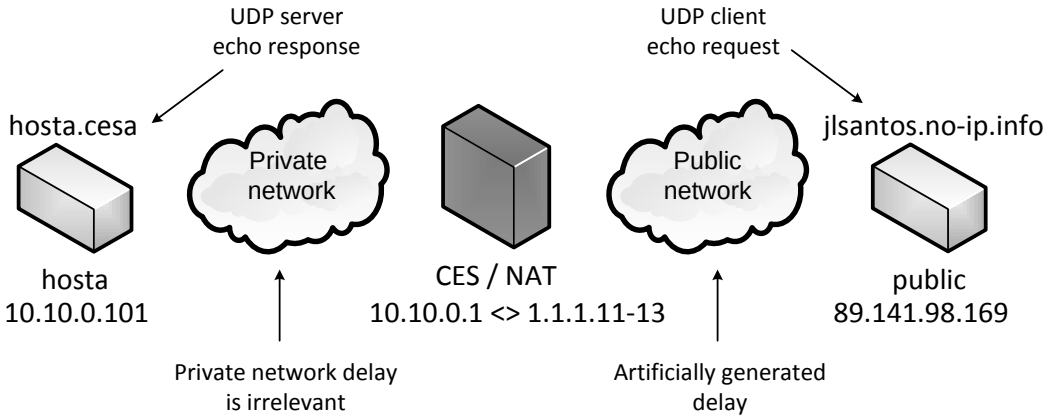


FIGURE 8.12 PERFORMANCE ANALYSIS – TESTING ENVIRONMENT

8.3.1 Impact of offered load and pool size with fixed delay

Figure 8.13 offers a graphical representation of the results obtained when submitting to test the previous scenario. The size of the pool is set to 3, 5 and 7 while the offered load follows an exponential distribution with an average inter-arrival time of 30, 50 and 70 new connections per second respectively. The network delay is set to 140 ms.

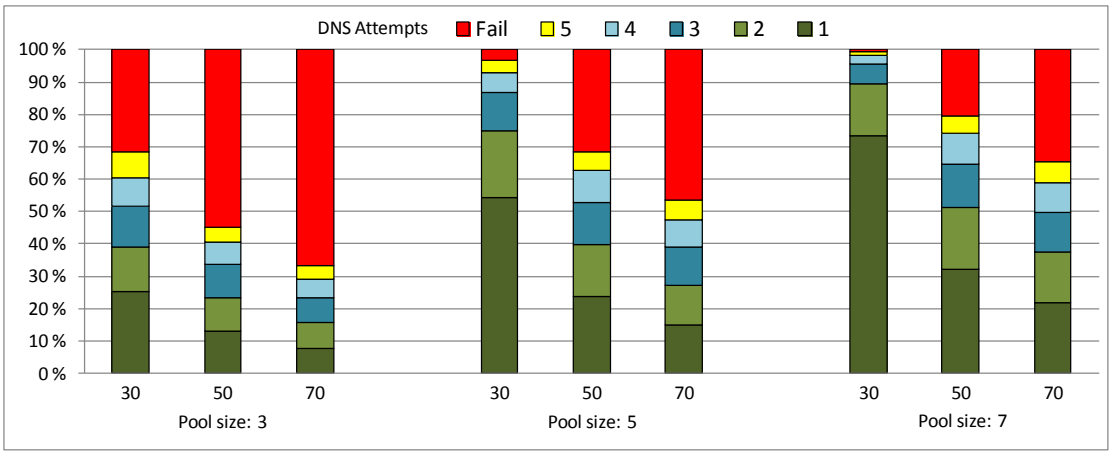


FIGURE 8.13 IMPACT OF OFFERED LOAD AND POOL SIZE WITH FIXED DELAY

[1] The value of 4 DNS retransmissions is typically found in Windows based Operating Systems.

8.3.2 Impact of network delay and pool size with fixed load

Figure 8.14 illustrates the results obtained when submitting to test different pool sizes and network delays. The size of the pool is set to 3, 5 and 7 addresses whereas the delay varies from 50, 80, 110 and 140 ms respectively. The offered load follows an exponential distribution with an average load of 60 new connections per second.

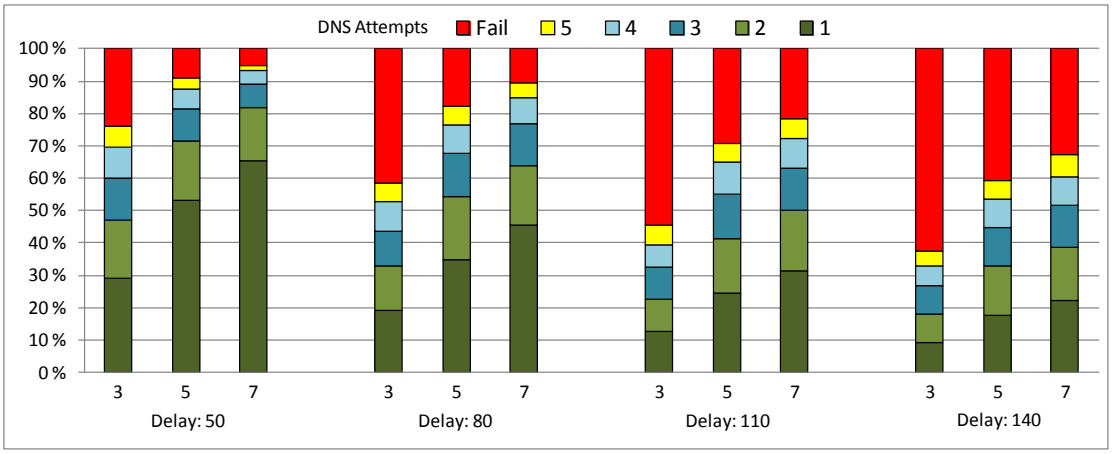


FIGURE 8.14 IMPACT OF DELAY AND POOL SIZE WITH FIXED OFFERED LOAD

8.3.3 Impact of packet loss with fixed delay

Figure 8.15 illustrates the results obtained when submitting to test different pool sizes, offered load and packet loss ratios with a network delay fixed to 110 ms. The size of the pool is set to 3, 5 and 7 addresses. The offered load follows an exponential distribution with an average load from 10 to 70 new connections per second. The evaluated packet loss ratios are 0.01%, 0.1%, 1% and ultimately 10%.

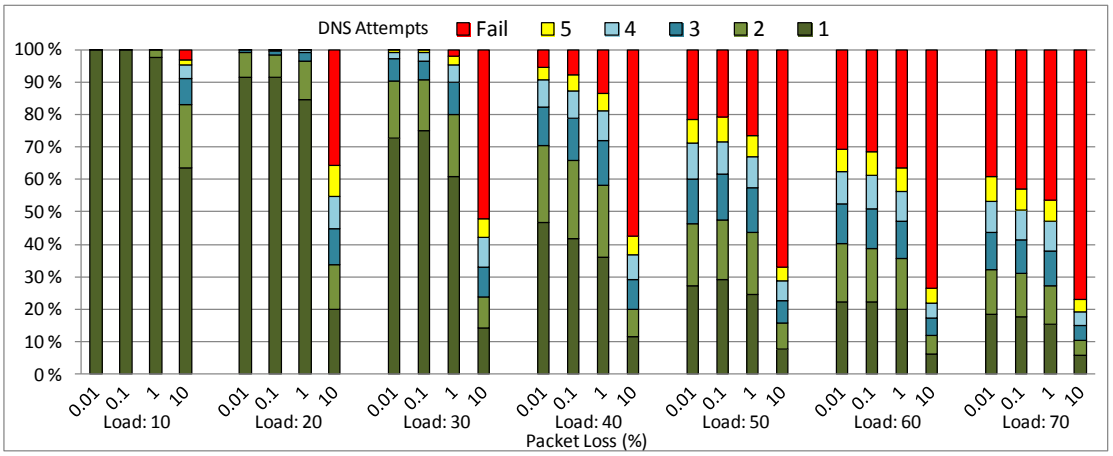


FIGURE 8.15 IMPACT OF PACKET LOSS WITH FIXED DELAY

8.3.3 Successful connections based on network delay and offered load

In Figure 8.16, Figure 8.17 and Figure 8.18 we represent the impact of the offered load in terms of successful connections measured for a pool of 3, 5 and 7 addresses. The variables analyzed are network delay and fixed values of offered load. As we anticipated, the figures reveal that the higher the offered load the lower is the percentage of successful connections.

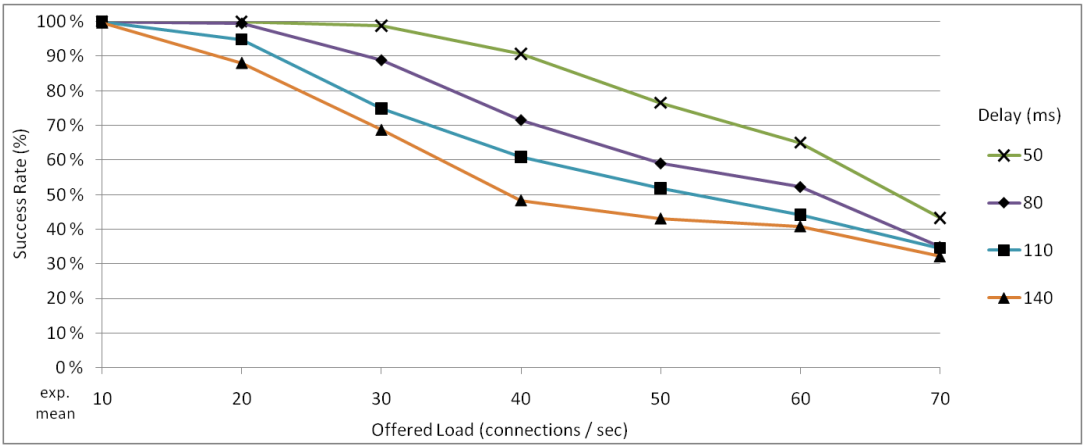


FIGURE 8.16 SUCCESSFUL CONNECTIONS FOR CIRCULAR POOL OF 3 ADDRESSES

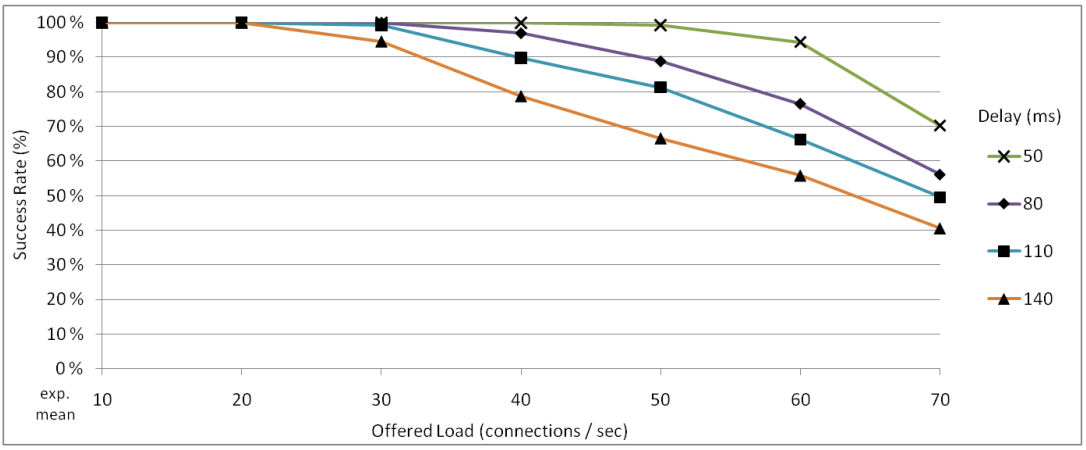


FIGURE 8.17 SUCCESSFUL CONNECTIONS FOR CIRCULAR POOL OF 5 ADDRESSES

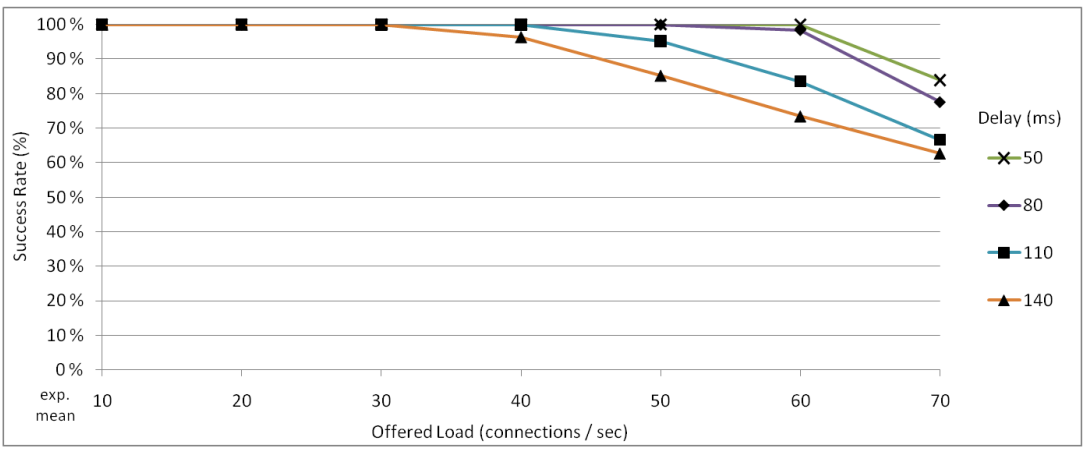


FIGURE 8.18 SUCCESSFUL CONNECTIONS FOR CIRCULAR POOL OF 7 ADDRESSES

Figure 8.19 represents the effects of the carried load as a function of the offered load for a pool of 5 addresses. There are three differentiated phases:

Ramp-up: The system operates under a light load with a high success rate. All the connections are accepted and processed.

Peak-capacity: The system operates at peak capacity reaching the maximum number of accepted connections. Some connections cannot be buffered and are dropped.

Fade-down: The system is affected by the high load. The buffer is greatly congested and many packets are dropped. Some of these will belong to the first data connection that shall release the allocated address in the circular pool. Failing to receive this packet, the IP address remains locked for the duration of the timeout. The performance of the system decreases due to the overall congestion and the lack of available addresses in the pool.

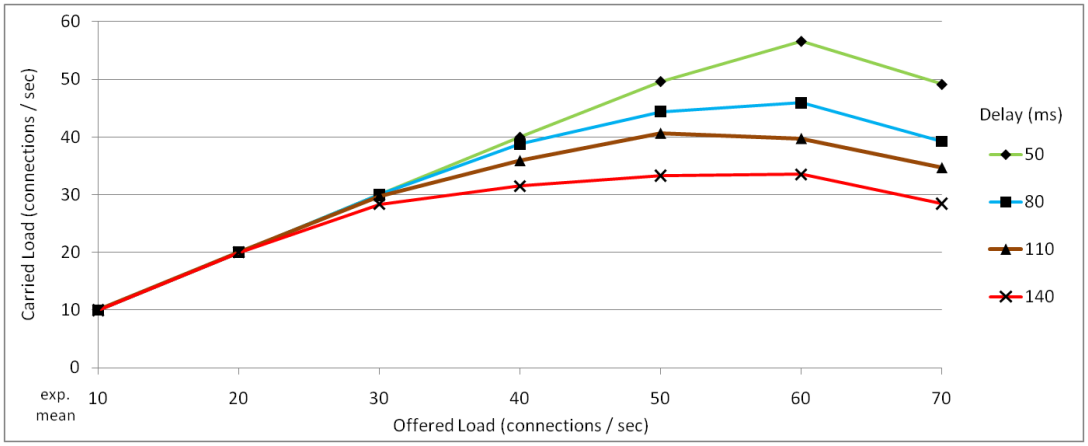


FIGURE 8.19 CARRIED LOAD VS OFFERED LOAD FOR CIRCULAR POOL OF 5 ADDRESSES

Figure 8.20 illustrates the comparison between the theoretical *Upper Bound*, the testing results and the theoretical values provided by the *B-Erlang* model for a pool of 5 addresses and a service time of 110 ms based on our results. The Upper Bound measures the theoretical maximum values the system is capable of whereas the B-Erlang model is derived from the Erlang distribution and indicates the blocking probability of a call for a certain amounts of resources without taking into account the reattempts typical of DNS.

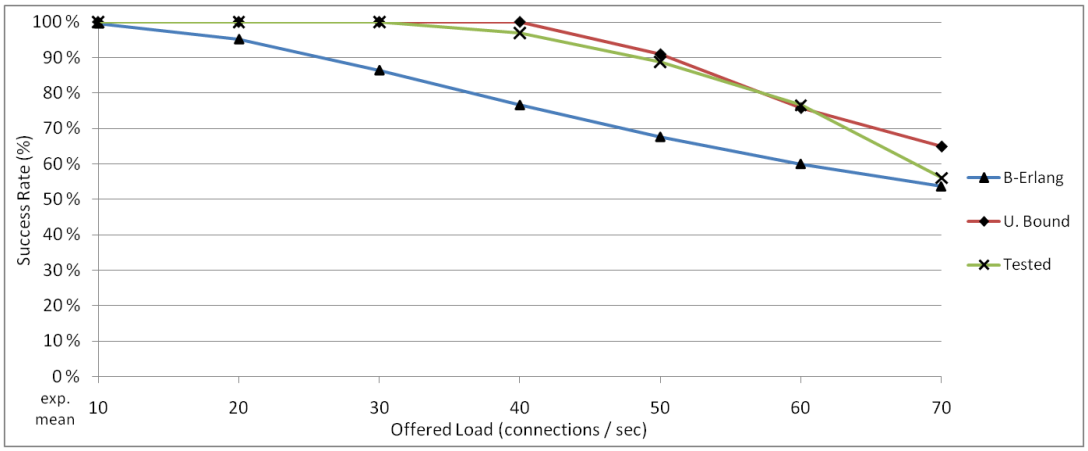


FIGURE 8.20 COMPARISON MODEL FOR CIRCULAR POOL OF 5 ADDRESSES

8.3.4 Summary of the Performance Analysis

Starting from Figure 8.13 to Figure 8.18 we have represented the inter-relation between the different variables that model the system; the offered load, the network delay and the pool size. We have also illustrated the influence of each one of them to the overall performance of the system and the prejudicial effects of the packet loss.

Then, in Figure 8.19 we represented the evolution of the system based on the offered load providing valuable information for a better understanding of the model.

Afterwards, in Figure 8.20 we presented a comparison of the theoretical values for the Upper Bound, the B-Erlang model and our own results. It is very important mentioning that neither of the theoretical models account for retransmissions when the connection is unsuccessful and therefore they assume *ideal* conditions. This being said, our testing results only seem to approximate the expected values when the DNS retransmissions are active. The reasons are mainly due to the testing environment and the actual implementation. With regard to the first one, the theoretical models assume perfect synchronization of messages, no packet loss and a fixed delay where applicable. The reality is that the computing power available introduced certain bottlenecks that resulted in an additional delay that limited the performance. In terms of implementation, the language chosen was Python that contributed to introduce yet new efficiency constraints.

Consider the scenario where an operator intends to release a VoIP service for its customers based on SIP. During the network provisioning phase, the operator has to

determine the amount of IP addresses that would need to be allocated for the public pool and how many users it would be possible to attend during the busy hour.

To that end, we have created a model based on the Erlang-B formula and a blocking probability of 0.1%. The offered load is calculated as **E = *h*λ (Erlang)**. The parameter *h* has been previously defined as the *service time*. The λ parameter corresponds with the arrival of calls per hour and it has been set to 1 call per hour and user.

Figure 8.20 illustrates the scalability of the system as a function of the available addresses and different service times of 100ms and 400ms. The results indicate that with a C Block address it would be possible to serve around 7.5M users.

It is very important to understand the implications of these results. For starters, the λ parameter indicated the number of calls per user during the busy hour, but the model only needs to perform the address allocation on incoming calls. In other words, either the λ parameter becomes 0.5 calls per user or the number of supported hosts doubles up to 15M users. Moreover, the significance of these results can be taken one step further if we consider that usually most of the calls happen within the same operator and those could benefit from large pools of private addresses. Under these circumstances, it is only necessary to allocate addresses from the pool when receiving a call from another operator, which makes the system even more efficient.

For these reasons, it is crucial to understand the requirements of the service that we are developing so that the network provisioning accounts for all these particularities that have great influence in the resulting scalability.

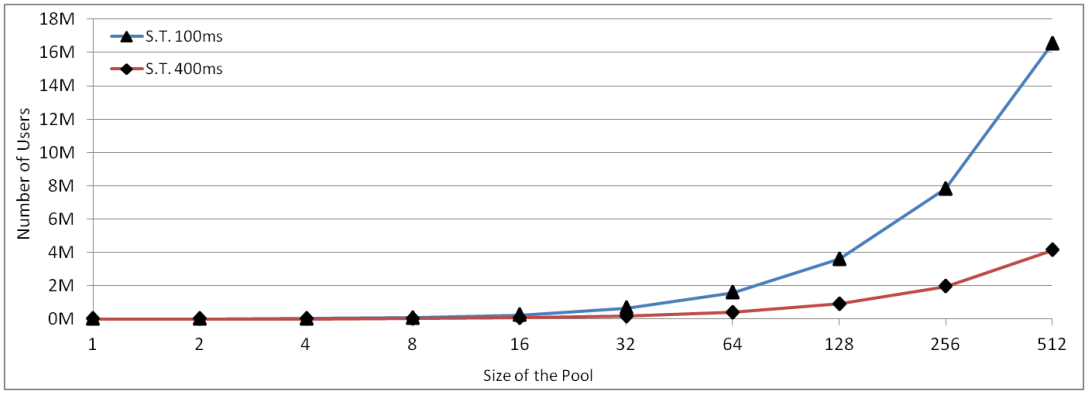


FIGURE 8.21 SYSTEM PERFORMANCE BASED ON SERVICE TIME

Figure 8.22 represents the efficiency of the system as a function of the pool size. The efficiency is calculated based on the carried load and the number of addresses

required with a blocking probability of 0.1%. Moreover, Figure 8.22 is complementary to Figure 8.21. The results indicate an asymptotic growth of the efficiency as long as the size of the pool is increased. The example reveals how the larger the pool the better is the efficiency. Furthermore, the marginal cost of an address decreases as long as the size of the pool increases. In practice the raise of the efficiency as a function of the number of addresses is steeper than the figure reveals since the DNS reattempts are not taken into account. The result is therefore an estimation of the lower bound.

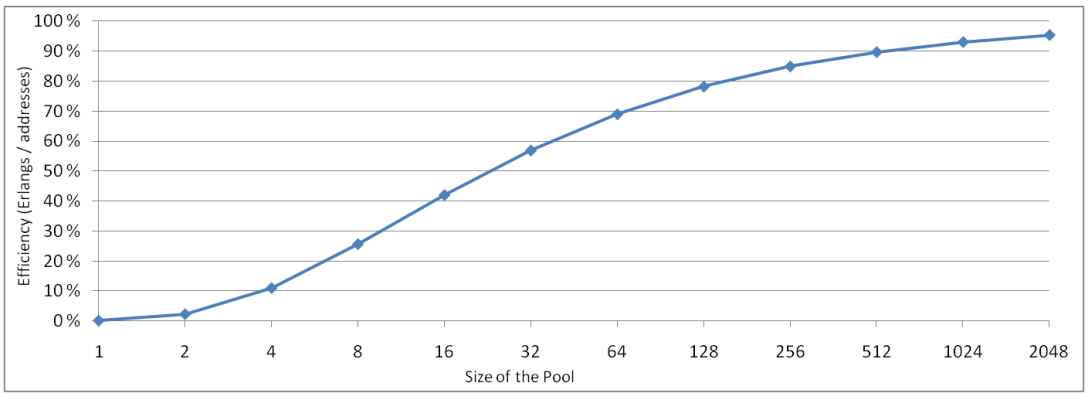


FIGURE 8.22 SYSTEM SCALABILITY BASED ON POOL SIZE

8.4 Additional Modifications in the Prototype

The current version of the CES prototype has improved the overall performance, stability and functionality compared with its predecessor. Below we present some of the most important changes introduced.

The DNS functionality was extended to allow interworking with legacy networks and additional DNS query types are supported. The prototype now supports traffic bursts due to buffering and implements a congestion control dropping packets under situations of heavy load with a probabilistic model. The previous sniffer module has been changed to a multi-threaded design with smart filtering of undesired packets. Several ALGs have been developed to guarantee end-to-end communication with most common protocols. A new module was coded to enable auto-configuration based on the network interfaces. The prototype supports remote access via console for state information visualization. Several bugs were detected and corrected that affected the stability of the system. The overall performance of the system has been greatly improved due to code optimizations and a customized version of Scapy for the packet

manipulation and forwarding. Finally, the prototype has been connected to the Internet and is now ready to be tested with real equipment.

Instructions for the installation and use of the demonstrator are available online in www.re2ee.org

8.5 Testing summary and Evaluation of Requirements

In this section we present a summary of the results obtained while testing the different protocols and applications. Table 8.13 indicates the application(s) and protocol(s) used in the hosts, the originator of the communication and the result of the operation. Regarding the result, there are three different possible values:

- Success: The test returned a positive result.
- ALG: The system uses an ALG for processing specific packets in order to guarantee end-to-end communication.
- Proxy: The system requires additional elements in the form of a proxy server to enable communication.

TABLE 8.13 – TESTING SUMMARY

Application in *hosta*	Application in *public*	Protocol(s)	Direction	Result
Netcat - client/server	Netcat - client/server	TCP & UDP	Both	Success
Ping – request	Ping – response	ICMP	Outgoing	~Success
Ping – response	Ping – request	ICMP	Incoming	~Success
–	Ping & Dig	Circular Pool DNS & ICMP	Incoming	Success
SSH – client/server	SSH – client/server	TCP	Both	Success
NTP client	NTP server	UDP	Outgoing	Success
Skype	Skype	TCP & UDP	Both	Success
Traceroute	Traceroute	ICMP Error	Both	ALG
HTTP client	HTTP server	HTTP / HTTPS	Outgoing	Success
HTTP server	HTTP client	HTTP / HTTPS	Incoming	Proxy
FTP client	FTP server	FTP Active	Outgoing	ALG
FTP client	FTP server	FTP Passive	Outgoing	Success
FTP server	FTP client	FTP Active	Incoming	Success
FTP server	FTP client	FTP Passive	Incoming	ALG

Attending to the functional requirements and design objectives hereby we present an analysis for each of these requirements. It is worth remembering that the major disadvantage of the solution was related to connectivity limitations, whereas the rest of these requirements did not appear to represent major problems.

Connectivity: The system reserves a public IP address based on an incoming DNS query containing the domain name of a host connected to the private network. State information is stored enabling a subsequent data packet to be forwarded to the intended recipient. The hosts can only be addressed by its domain name and are not directly reachable by a public IP address.

Flexibility: The system is designed to easily accommodate new protocols whenever they are developed. Two Application Layer Gateways have been developed to overcome the connectivity issues manifested by FTP and ICMP protocols when performing address translation operations. In addition, HTTP and HTTP(S) traffic is forwarded to a HTTP-Proxy located in the private network that fetches the content from the intended recipient and delivers it to the remote host. This solution was inspired by the concepts of "*Unique Global IP*" and "*Domain Based Packet Forwarding*".

Scalability: The performance analysis introduced in Section 8.3 illustrates how both the design and the implementation were successful. The system proved highly efficient accepting a large number of *new incoming connections* in spite of the limited size of the address pool. The number of hosts allocated by the *Private Realm Gateway* does not have negative effects on the system. In addition, the solution not only does not aggravate the address exhaustion, but contributes to alleviating the problem.

Deployment: The model provides a transparent framework and does not require changes in either the network topology or any of the hosts; protocols are not modified either. Furthermore, the system is potentially deployable on its own since it provides immediate benefits to all the hosts located in the private network.

Security and Trust: In this aspect, we were able to identify up to four different scenarios that could lead to DoS/DDoS attacks therefore affecting the security of the system. To that end, we proposed several methods of protection against these attacks. We also introduced a mechanism to regulate the load of the system to mitigate

possible DoS/DDoS attacks. Regarding the security of the hosts, it is possible to add firewall capabilities with specific rules to determine whether a packet is allowed to be forwarded or not. In addition the hosts are not directly exposed to the public network and become only reachable after performing a name resolution operation. As a consequence, hosts remain *hidden* from possible attackers. The system is able to monitor suspicious traffic flows that can be used for future reporting to a *Trust Management System*. In this sense, it would be possible to establish honey pots in order to trap malicious hosts and report them to an Internet wide Trust Management System for further actions.

Table 8.14 summarizes the evaluation of the requirements.

TABLE 8.14 –EVALUATION OF REQUIREMENTS

REQUIREMENTS	NOTES
Connectivity	The requirements are met. Reachability problem is solved.
Flexibility	The requirements are met. The system behaves adequately.
Scalability	The requirements are met. The system is highly efficient.
Deployment	The requirements are met. Potentially deployable.
Security and Trust	The requirements are met. Future work on the topic is needed.

9. Conclusions

This Master's thesis was originally intended to extend the previous research on Customer Edge Switching (CES). The main focus was to provide interworking of CES networks with public legacy networks. Considering the private realms of addresses defined for the CES architecture, connecting these private hosts to public networks would require address translation operations.

The reachability problem that accompanies address translation was one of the major obstacles. Furthermore, there were additional requirements that hindered the development of the project. These requirements were mostly related to efficient address allocation, connectivity and transparency. The concept we presented receives the name of Private Realm Gateway (PRGW).

We proposed three different solutions. After considering their advantages and disadvantages, the scale tipped in favor of the Circular Pool model. The major disadvantages of the Circular Pool were related to vulnerabilities and possibilities of denial of service from malicious users or botnets. A brief analysis of security was presented to motivate future research.

Additionally, some connectivity issues were detected and solved by developing specific ALGs for ICMP and FTP protocols. In the other cases, regarding web and HTTP(S) protocols, end-to-end communication was granted with the inclusion of an HTTP proxy in the architecture. As of the final version of the implementation, all the three designs we proposed have been used. The *Circular Pool* is used for handling incoming connection whereas a combination of *Unique Global IP* and *Domain Based Packet Forwarding* has been used for incoming HTTP(S) traffic. As a summary, the address translation operation follows the "*address and port-dependent mapping and filtering*" as it is defined by the RFC 4787 [3].

The actual implementation was integrated within the previous CES prototype. Nevertheless, the functionality introduced by PRGW is completely independent and suitable to be shipped as standalone package. Regarding the magnitude of the implementation, the original application consisted of 740 lines of Python code that were extended to around 3220 lines upon finishing this project.

The evaluation proved the concept and the implementation highly successful. Not only because all the network protocols and the applications were positively tested but also because of the scalability and performance of the system. The tests revealed how the PRGW model, by making use of a rather limited pool of addresses, is able to establish a very large number of incoming connections per second in the long run. The design requirements were also evaluated and submitted to analysis. We concluded that these requirements were successfully met.

Our goal was to create a framework for connecting wireless devices to the Internet attending to very specific requirements. We have shown that PRGW succeeds for this use case. However, we are aware of the limitations of the circular pool solution for the case of connecting heavy duty servers with a very high level of new flow arrivals per second particularly when the connection would be processed by the circular pool.

The deployment of PRGW is transparent to the network and not only introduces benefits regarding legacy communications, but also promotes the adoption of the CETP protocol leading towards security and end-to-end trust. Furthermore, a PRGW does not require anything in particular from the network since it behaves like any other connected device, which makes it ideal for progressive deployments.

All in all, the Circular Pool appears as a very promising solution not only by helping to solve the reachability problem but also by contributing to solving the address exhaustion. Strongly motivated by transparent operations and not requiring any further changes in hosts or network, this new version of CES, could play an essential role in future acceptance and standardization.

9.1 Future Work

This section introduces some important topics that were not covered by this research or were considered out of the scope.

Security: The CES prototype is ready to support security based on dropped packets. Attack detection as well as proactive and reactive mechanisms must be implemented to provide higher degree of security in the system.

NAT Traversal Protocols: Analyzing the impact of STUN/TURN/ICE operations within the PRGW model and determine if there is still a need to use these protocols. Conduct further testing where an end device is located behind a NAT using STUN/TURN/ICE and the other is behind a PRGW.

Multihoming and mobility: The current architecture does not provide support for roaming clients. Synchronization of the state information between CES devices is required. Redundancy of CES is a collateral, but necessary, effect of these changes.

Nested CES with legacy capabilities: Study the possibility of allocating multiple CES devices in a hierarchical pattern. Analyzing the impact on CES and legacy communication and adapting the model appears to be a challenging task.

Control plane and data plane separation: Implement a dedicated data plane for fast forwarding of packets in a more efficient programming language, such as C or C++. The control plane would be only responsible for establishing new mapping and ALG implementations.

References

[1] P. Albitz and C. Liu, "*DNS and BIND,"* 5th edition, May 2006.

[2] M. Allman, V. Paxson and E. Blanton, "*TCP Congestion Control,*" RFC 5681, Sep. 2009.

[3] F. Audet and C. Jennings, "*Network Address Translation (NAT) Behavioral Requirements Unicast UDP,*" RFC 4787, Jan. 2007.

[4] R. Braden, "*Requirements for Internet Hosts – Communication Layers,*" RFC 1122, Oct. 1989.

[5] S. Deering and R. Hinden, "*Internet Protocol, Version 6 (IPv6) Specification,*" RFC 2460, Dec. 1998.

[6] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach and T. Berners-Lee, "*Hypertext Transfer Protocol - HTTP/1.1,*" RFC 2616, June 1999.

[7] A. Freier, P. Karlton and P. Kocher, "*The Secure Sockets Layer (SSL) Protocol Version 3.0,*" RFC 6101, Aug. 2011.

[8] S. Guha and N. Daswani, "*An Experimental Study of the Skype Peer-to-Peer VoIP System*", Cornell University, Dec. 2005. [Retrieved on 13.03.2012] Available: http://dspace.library.cornell.edu/handle/1813/5711

[9] ICANN, "*Available Pool of Unallocated IPv4 Internet Addresses Now Completely Emptied,*" Feb. 2011. [Retrieved on 13.03.2012] Available: http://www.icann.org/en/news/releases/release-03feb11-en.pdf

[10] Information Sciences Institute University of Southern California for DARPA "*Internet Protocol, DARPA Internet Program Protocol Specification,*" RFC 791, Sep. 1981.

[11] ITU-T ICT Developments 2009. Free statistics. [Retrieved on 13.03.2012] Available: http://www.itu.int/ITU-D/ict/statistics

[12] R. Kantola, N. Beijar, Z. Yan and M. Pahlevan, "*Customer Edge Traversal Protocol (CETP),*" Aalto University, Department of Communications and Networking. Sep. 2012. Work in progress. Available: http://www.re2ee.org/

[13] R. Kantola "*Implementing Trust-to-Trust with Customer Edge Switching,"* AMCA in connection with AINA 2010. Aalto University, Department of Communications and Networking. Available: http://www.re2ee.org/

[14] R. Kantola, M. Luoma and J. Manner, "*Future Internet is by Ethernet,*" World Computer Congress, Brisbane, Australia. Aalto University, Department of Communications and Networking. Sep. 2010. Available: http://www.re2ee.org/

[15] P. Leppäaho, "*Design of Application Layer Gateways for Collaborative Firewalls,*" M.Sc. Thesis. Aalto University, Department of Communications and Networking. May 2012.

[16] J. Llorente, "*Testbed for Customer Edge Switching,"* Special Assignment. Aalto University, Department of Communications and Networking, Oct. 2012.

[17] R. Mahy, P. Matthews and J. Rosenberg, "*Traversal Using Relays around NAT (TURN): Relay Extensions to Session Traversal Utilities for NAT (STUN),*" RFC 5766, April 2010.

[18] J. Mercier, "*Skype numerology,*" [Retrieved on 13.03.2012] Available: http://skypenumerology.blogspot.com/

[19] K. Nichol, S. Blake, F. Baker and D. Black, "*Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers*," RFC 2474, Dec. 1998.

[20] NRO Regional Internet Registries "*Free Pool of IPv4 Address Space Depleted,*" Feb. 2011. [Retrieved on 13.03.2012] Available: http://www.nro.net/news/ipv4-free-pool-depleted

[21] J. Postel and J. Reynolds, "*File Transfer Protocol (FTP),*" RFC 959, Oct. 1985.

[22] J. Postel, "*Internet Control Message Protocol, DARPA Internet Program Protocol Specification,*" RFC 792, Sep. 1981.

[23] J. Postel, "*User Datagram Protocol,*" RFC 768, Aug. 1980.

[24] K. Ramakrishnan, S. Floyd and D. Black, "*The Addition of Explicit Congestion Notification (ECN) to IP,*" RFC 3168, Sep. 2001.

[25] Y. Rekhter and T. Li, "*An Architecture for IP Address Allocation with CIDR,*" RFC 1518, Sep. 1993.

[26] Y. Rekhter, B. Moskowitz, D. Karrenberg, G. J. de Groot and E. Lear, "*Address Allocation for Private Internets,*" RFC 1918, Feb. 1996.

[27] E. Rescorla, "*HTTP over TLS,*" RFC 2818, May 2000.

[28] J. Rosenberg, R. Mahy and P. Matthews, "*Session Traversal Utilities for NAT (STUN),*" RFC 5389, Oct. 2008.

[29] J. Rosenberg, "*Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols,*" RFC 5245, April 2010.

[30] Skype. [Retrieved on 13.03.2012] Available: http://www.skype.com

[31] P. Srisuresh and K. Egevang, "*Traditional IP Network Address Translator (Traditional NAT),*" RFC 3022, Jan. 2001.

[32] S. Turner and T. Polk, "*Prohibiting Secure Sockets Layer (SSL) Version 2.0,*" RFC 6176, March 2011.

[33] UPnP Forum, "*UPnP Specifications,*" [Retrieved on 20.04.2012]. Available: http://upnp.org/sdcps-and-certification/standards/

[34] L. Virtanen "*Communicating Globally Using Private IP Addresses,*" M.Sc. Thesis, Helsinki University of Technology, Department of Communications and Networking. May 2009.

[35] Z. Yan and R. Kantola, "*Unwanted Traffic Control via Global Trust Management,*" IEEE TrustCom 2011, pp. 647 – 654. Changsha, China, Nov. 2011.

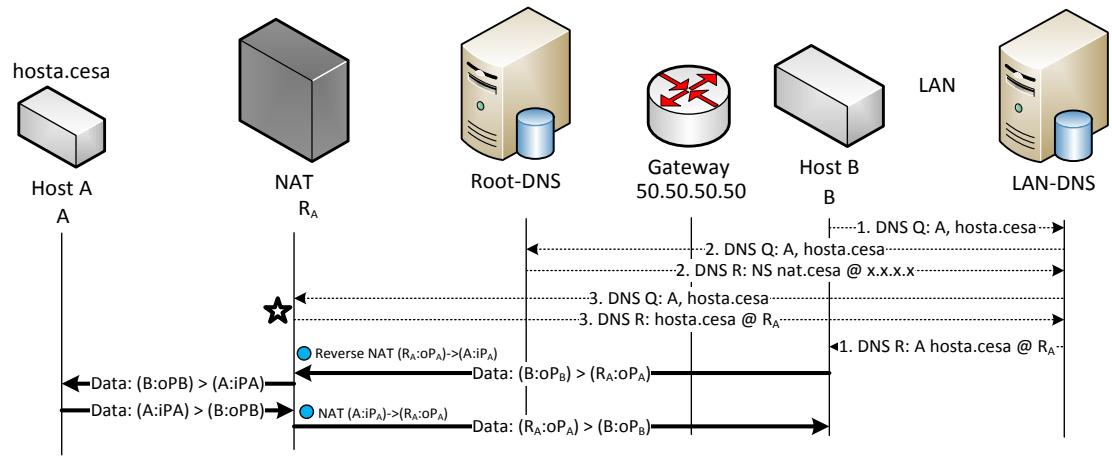
Appendix A – Extended DNS Scenarios

This appendix introduces further research in addition to Section 5.3 - *Incoming Connections*. Continuing with the study of incoming connections on the Internet model, this appendix illustrates two different scenarios. First a DNS server is located in a private network together with the originating host. Then the DNS resolution process is offloaded from the NAT device to an external server.

Although the name resolution process is similar for both cases returning always the same result, the information that can be inferred varies from one to another model due to the location of the DNS server. The following scenarios represent a successful data connection due to extended forwarding configuration enabling incoming packets to traverse the NAT device.

Private DNS server in private network

Consider that both the originating host and the DNS server are located in a private realm with Internet connectivity provided via a NAT device. Attending to the DNS resolution process, the local DNS server will attempt to resolve iteratively the domains queried. Eventually, the source IP address of the DNS query received in the NAT is the same as for the first data packet. The scenario is depicted in Figure A.1.



A: Private IP address of host A	B: Public IP address of host B
oP_A: Outward port of NAT (public port)	oP_B: Outward port of B (public port)
R_A: Unique global IP address of NAT for host A	

FIGURE A.1 INCOMING CONNECTION WITH PRIVATE DNS SERVER

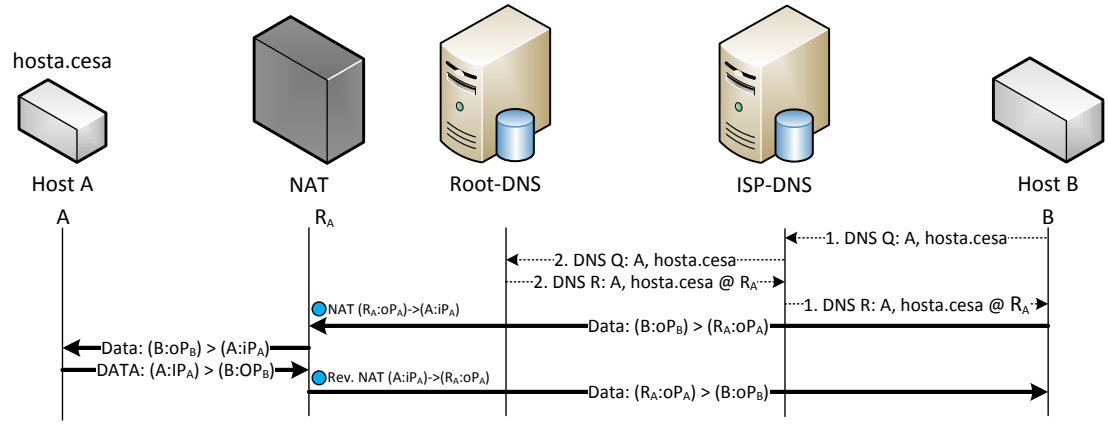
Note that this scenario is also applicable to a single host directly connected to the Internet running a DNS server on its own device.

The advantages and disadvantages are discussed as follows:

- Advantages: This model would allow establishing mapping on the fly for the next data packet enabling adequate traversal of packets through the NAT.
- Disadvantages: Due to the DNS architecture and the operation mode it would not be possible to assure with all certainty that the originating IP address of the DNS query is the same as for the data connection.
- Summary: The model could make certain contributions but since it is not entirely reliable we cannot use it.

DNS zone records offloaded to an external server

Consider for the following scenario the recipient NAT device is no longer authoritative for the zone "*.cesa*". The zone records have been delegated to another DNS server that now handles all the domain resolution. This is usually the case where individual hosts maintain a particular domain on the Internet that is updated upon detecting a change on their public IP address by using DDNS. Figure A.2 illustrates the scenario proposed.



A: Private IP address of host A	B: Public IP address of host B
oP_A: Outward port of NAT (public port)	oP_B: Outward port of B (public port)
R_A: Unique global IP address of NAT for host A	

FIGURE A.2 INCOMING CONNECTION WITH DNS OFFLOAD

- Advantages: Offloading the DNS processing in a NAT device reduces the system complexity. Use of standard applications, e.g. *Bind* to process such messages.
- Disadvantages: Unawareness of name resolution for private hosts located behind NAT. Unable to use incoming domain resolution for data forwarding.
- Summary: The model does make any contribution and therefore is discarded without further considerations.

Appendix B – Network Protocol Tests

This appendix contains the test cases previously explained in Section 8.1.1 and seeks to evaluate the behavior of the network prototype when submitted to basic operations involving TCP, UDP and ICMP protocols. The topology of the scenario is illustrated in Figure 8.1. Testing result consists of command line output of both private and remote hosts as well as the forwarding table status in the CES prototype.

TCP and UDP protocols are tested with the same application, *Netcat - The TCP/IP Swiss Army Knife*. This particular application provides both client and server functionality thus enabling the user to start a client-to-server communication with the same suite. The messages input on the client side are forwarded and displayed in the server side. The port selected by the server is 12345 for all tests.

Outgoing TCP connection

During this first test, we attempt to establish an outgoing TCP connection between *hosta* and *public*. The device *public* binds locally a TCP socket and remains waiting for an incoming connection. When the connection takes place, the server will display the message sent by the client.

Originating *hosta* initiates a connection towards *public* and delivers a message. The packet then is sent to the CES device that performs an address translation and forwards it to *public*. The result of the operation is displayed below.

Private Host:

```
tester@hosta:~$ nc 89.141.98.169 12345
Sending a message behind CES
to a public host on the Internet
via TCP
^C
tester@hosta:~$
```

Remote Host:

```
tester@public:~$ nc -l 12345
Sending a message behind CES
to a public host on the Internet
via TCP
tester@public:~$
```

CES:

TABLE B.1 – NC TCP OUTGOING CONNECTION

NAT TABLE LEGACY STATUS								
LOCAL		OUTBOUND		REMOTE		Prot.	Tout.	Status
IP	Port	IP	Port	IP	Port			
10.10.0.101	33709	1.1.1.11	33709	89.141.98.169	12345	TCP	1800	A

Additional notes: The operation is successful as it is represented in the output generated on the terminals and the forwarding table. The entry represented in the previous table appears as *active* because TCP requires connection establishment and acknowledging of data packets.

Incoming TCP connection

This test attempts to establish an incoming TCP connection between *public* and *hosta*. The process is similar to the previous one with the exception of the name resolution issued by *public*. As a result, an IP address is allocated from the circular pool. Address translation and packet forwarding follows the same fashion as before. The result of the operation is displayed below.

Remote Host:

```
tester@public:~$ nc hosta.cesa 12345
This is remote host sending
a message to private Host-A
via TCP
^C
tester@public:~$
```

Private Host:

```
tester@hosta:~$ nc -l 12345
This is remote host sending
a message to private Host-A
via TCP
tester@hosta:~$
```

CES:

TABLE B.2 – NC TCP INCOMING CONNECTION

NAT TABLE LEGACY STATUS								
LOCAL		OUTBOUND		REMOTE		PROT.	TOUT.	STATUS
IP	PORT	IP	PORT	IP	PORT			
10.10.0.101	12345	1.1.1.11	12345	89.141.98.169	47500	TCP	1800	A

Additional notes: Likewise in the previous case the operation succeeds resulting in an a*ctive* entry in the forwarding table.

Outgoing UDP connection

This test attempts to send a UDP message from *hosta* to *public*. The operation is identical to "Outgoing TCP connection" and follows the same criteria. The result of the operation is displayed below.

Private Host:

```
tester@hosta:~$ nc -u 89.141.98.169 12345
Sending a message behind CES
to a public host on the Internet
via UDP
^C
tester@hosta:~$
```

Remote Host:

```
tester@public:~$ nc -u -l 12345
Sending a message behind CES
to a public host on the Internet
via UDP
tester@public:~$
```

CES:

TABLE B.3 – NC UDP OUTGOING CONNECTION

NAT TABLE LEGACY STATUS								
LOCAL		OUTBOUND		REMOTE		Prot.	Tout.	Status
IP	Port	IP	Port	IP	Port			
10.10.0.101	40275	1.1.1.11	40275	89.141.98.169	12345	UDP	60	O

Additional notes: The operation is successful as it is represented in the output generated on the terminals and the forwarding table. The entry represented in the forwarding table appears as *outgoing* because there has not been a response from *public*. All the traffic so far is unidirectional.

Incoming UDP connection

During this forth test, we attempt to send a UDP message from *public* to *hosta*. The operation is identical than with TCP and follows the same criteria. The result of the operation is displayed below.

Remote Host:

```
tester@public:~$ nc -u hosta.cesa 12345
This is remote host sending
a message to private host-A
via UDP
tester@public:~$
```

Private Host:

```
tester@hosta:~$ nc -u -l 12345
This is remote host sending
a message to private host-A
via UDP
tester@hosta:~$
```

CES:

TABLE B.4 – NC UDP INCOMING CONNECTION

NAT TABLE LEGACY STATUS								
LOCAL		OUTBOUND		REMOTE		Prot.	Tout.	Status
IP	Port	IP	Port	IP	Port			
10.10.0.101	12345	1.1.1.11	12345	89.141.98.169	51354	UDP	60	I

Additional notes: Likewise in the previous case the operation succeeds resulting in an *incoming* entry in the forwarding table.

The ICMP protocol is tested with the ping command. This application sends an ICMP echo request that is answered back from the destination with an ICMP echo response. This application is widely used for connectivity test, providing with valuable information of current network conditions.

Outgoing ICMP

This scenario tests the ICMP functionality for outgoing echo requests. In this case we will first attempt to ping the domain *jlsantos.no-ip.info* and then the IP address associated with this domain *89.141.98.169*. The result of the operation is displayed below.

Private Host:

```
tester@hosta:~$ ping jlsantos.no-ip.info
PING jlsantos.no-ip.info (89.141.98.169) 56(84) bytes of data.
64 bytes from 89.141.98.169.dyn.user.ono.com (89.141.98.169):
icmp seq=1 ttl=63 time=69.6 ms
64 bytes from 89.141.98.169.dyn.user.ono.com (89.141.98.169):
icmp_seq=2 ttl=63 time=9.11 ms
^C
--- jlsantos.no-ip.info ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 9.119/39.367/69.616/30.249 ms

tester@hosta:~$ ping 89.141.98.169
PING 89.141.98.169 (89.141.98.169) 56(84) bytes of data.
64 bytes from 89.141.98.169: icmp_seq=1 ttl=63 time=7.50 ms
64 bytes from 89.141.98.169: icmp_seq=2 ttl=63 time=11.1 ms
^C
--- 89.141.98.169 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1005ms
rtt min/avg/max/mdev = 7.507/9.341/11.175/1.834 ms
```

CES:

TABLE B.5 – PING ICMP OUTGOING CONNECTION

NAT TABLE LEGACY STATUS								
LOCAL		OUTBOUND		REMOTE		Prot.	Tout.	Status
IP	Port	IP	Port	IP	Port			
10.10.0.101	33286	1.1.1.11	33286	89.141.98.169	33286		60	A
10.10.0.101	33542	1.1.1.11	33542	89.141.98.169	33542	ICMP	60	A

Additional notes: Based on the output produced on the terminal and the forwarding table, the operation is successful. The entry appears as *active* indicating bidirectional communication motivated by the echo request/response.

Incoming ICMP

This scenario tests the ICMP functionality for incoming echo requests. In this case we will first attempt to ping the domain "*hosta.cesa"* and then directly with public IP address associated with this domain for the first query *"1.1.1.13"*. The following lines display the console information on the hosts and the forwarding table in CES.

Remote Host:

```
tester@public:~$ ping hosta.cesa
PING hosta.cesa (1.1.1.13) 56(84) bytes of data.
64 bytes from 1.1.1.13: icmp_seq=1 ttl=63 time=20.7 ms
--- hosta.cesa ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1067ms
rtt min/avg/max/mdev = 20.751 /20.751 /20.751 /0.000 ms

tester@public:~$ ping 1.1.1.13
PING 1.1.1.13 (1.1.1.13) 56(84) bytes of data.
^C
--- 1.1.1.13 ping statistics ---
15 packets transmitted, 0 received, 100% packet loss, time 16126ms
```

CES:

TABLE B.6 – PING ICMP INCOMING CONNECTION

NAT TABLE LEGACY STATUS								
LOCAL		OUTBOUND		REMOTE		Prot.	Tout.	Status
IP	Port	IP	Port	IP	Port			
10.10.0.101	17158	1.1.1.11	17158	89.141.98.169	17158	ICMP	60	A

Additional notes: The operation is partly successful, allowing only the communication when a domain resolution takes place. Consequently, the attempt of *pinging* directly a public IP address of the circular pool will always fail since it is a feature of the circular pool itself. These packets are dropped by the CES. For the successful connection, the entry appears as *active* indicating bidirectional communication motivated by the echo request/response.

On the other hand, we have detected some scenarios where an ICMP *connection* is prone to fail. The reasons are due to the simplistic way of creating the mapping in the forwarding table. The major difference of ICMP and TCP or UDP with respect to NATs is that ICMP does not contain a 16 bits field to indicate a source or destination port. As a consequence, the mapping created in our prototype is slightly different and uses a combination of the field type and code to establish this mapping.

Considering that an echo request uses the *type/code* values 8/0, and the echo response 0/0 the limiting factor becomes quite clear. The current prototype implementation cannot process more than "*N"* ICMP connections at a time. The *N* variable corresponds with the size of the circular pool allocated in the CES.

The risks of having more than N ICMP connections through the PRGW are the following:

Mapping overlap: Due to the reduced identification and differentiation of ICMP packets, it is possible that a new connection is thought as ongoing connection thus reusing the previous mapping or overwriting it with the values.

Packet misrouting: As a consequence of the previous case, a modification of the state information or failure to distinguish between an ongoing and a new connection may lead to misrouting of packets in the private network.

This is a small issue that requires a solution although we have decided not to pursue any further actions.

In the short term, it is possible to limit the amount of ongoing ICMP connections, which translates into packet dropping when the N factor is achieved.

In the long term, it would be interesting to study further the ICMP protocol and the implementation that most common operating system make of it. To that end, a new ALG could be developed enabling transparent end-to-end ICMP connections without specific limitations.

Appendix C – Skype Test

Back in 2001, Niklas Zennström co-founded with Janus Friis Kazaa, a peer-to-peer file sharing application, two years later, in 2003, Skype was released. Skype is an application that integrates VoIP, instant messaging and file transfer functionalities [30]. It is available for most platforms and operating systems as well as computers and mobile devices. Because the excellent user-experience provided, due to the ability of operate behind NATs and firewall, by of the end of 2009 there were 521 million registered users. As of March 2012, the peak of simultaneous connected users is around 37 million and is expected to keep growing every day. [18]

Despite the popularity little is known about its proprietary protocols and network. Based on the similarities with Kazaa, a connection setup and usage of "supernodes" have been discovered. The architecture of Skype is based on two layers, supernodes and ordinary nodes. Supernodes are connected between them by an overlay network whereas ordinary nodes are typically connected to a small set of supernodes. Ordinary nodes send control information over the peer-to-peer network maintained by the supernodes. Upon accepting a session, end devices will attempt to establish a direct end-to-end connection by using a STUN-like protocol, falling back to TURN-like in case of failure. In case of the latter, the relay is a public reachable supernode. Depending on the network conditions and architecture, an ordinary node can become a supernode in a matter of minutes and relay traffic from other users. [8]

The information displayed below represents how the implementation of the network prototype is able to cope with the characteristics of a peer-to-peer network. The following table displays how after starting Skype, the application tries to contact other hosts stored in the host-cache via UDP and establish a communication with several supernodes via TCP. The forwarding table at this moment is represented below.

CES:

TABLE C.1 – SKYPE INITIAL CONNECTIONS

NAT TABLE LEGACY STATUS								
LOCAL		OUTBOUND		REMOTE		Prot.	Tout.	Status
IP	Port	IP	Port	IP	Port			
10.10.0.101	1602	1.1.1.11	1602	130.204.236.68	37888	UDP	60	A
10.10.0.101	1602	1.1.1.11	1602	114.47.194.118	3382	UDP	60	A
10.10.0.101	44940	1.1.1.11	44940	83.49.166.168	31512	TCP	1800	A
10.10.0.101	1602	1.1.1.11	1602	95.25.115.84	46590	UDP	60	A
10.10.0.101	1602	1.1.1.11	1602	75.253.228.147	12788	UDP	60	A
10.10.0.101	1602	1.1.1.11	1602	95.65.21.211	9996	UDP	60	A
10.10.0.101	1602	1.1.1.11	1602	83.143.144.17	36674	UDP	60	A
10.10.0.101	1602	1.1.1.11	1602	122.215.20.147	65105	UDP	60	A
10.10.0.101	1602	1.1.1.11	1602	24.62.188.73	443	UDP	60	A
10.10.0.101	1602	1.1.1.11	1602	121.95.212.45	22643	UDP	60	A
10.10.0.101	54488	1.1.1.11	54488	62.43.101.57	21739	TCP	1800	A
10.10.0.101	1602	1.1.1.11	1602	81.232.125.44	19185	UDP	60	A
10.10.0.101	1602	1.1.1.11	1602	125.13.52.64	38158	UDP	60	A
10.10.0.101	1602	1.1.1.11	1602	24.34.45.45	60236	UDP	60	A
10.10.0.101	1602	1.1.1.11	1602	195.113.61.48	44905	UDP	60	A
10.10.0.101	60057	1.1.1.11	60057	204.9.163.247	80	TCP	12	A
10.10.0.101	1602	1.1.1.11	1602	217.208.205.94	39652	UDP	60	A
10.10.0.101	1602	1.1.1.11	1602	205.146.120.75	52504	UDP	60	A
10.10.0.101	1602	1.1.1.11	1602	123.110.181.41	62784	UDP	60	A
10.10.0.101	1602	1.1.1.11	1602	201.237.91.126	41729	UDP	60	A
10.10.0.101	1602	1.1.1.11	1602	146.247.214.130	28526	UDP	60	A
10.10.0.101	1602	1.1.1.11	1602	46.118.16.135	25918	UDP	60	A
10.10.0.101	1602	1.1.1.11	1602	46.150.246.80	40764	UDP	60	A
10.10.0.101	1602	1.1.1.11	1602	46.241.47.166	6307	UDP	60	A
10.10.0.101	1602	1.1.1.11	1602	46.150.252.202	11573	UDP	60	A
10.10.0.101	1602	1.1.1.11	1602	208.88.186.11	34027	UDP	60	A
10.10.0.101	1602	1.1.1.11	1602	94.243.208.125	38701	UDP	60	A
10.10.0.101	1602	1.1.1.11	1602	149.5.45.4	43038	UDP	60	A
10.10.0.101	1602	1.1.1.11	1602	129.194.31.186	54508	UDP	60	A
10.10.0.101	1602	1.1.1.11	1602	76.117.188.105	23802	UDP	60	A
10.10.0.101	1602	1.1.1.11	1602	149.13.32.15	13392	UDP	60	A
10.10.0.101	1602	1.1.1.11	1602	200.127.86.53	26357	UDP	60	A
10.10.0.101	**48018**	**1.1.1.11**	**48018**	**217.208.205.94**	**39652**	**TCP**	**1800**	**A**
10.10.0.101	1602	1.1.1.11	1602	31.147.130.15	30892	UDP	60	A
10.10.0.101	1602	1.1.1.11	1602	24.99.190.124	443	UDP	60	A
10.10.0.101	1602	1.1.1.11	1602	151.77.164.43	31405	UDP	60	A
10.10.0.101	1602	1.1.1.11	1602	188.27.212.127	39975	UDP	60	A

After some minutes of inactivity, old entries are cleared from the forwarding table. As a result, only one connection with a supernode remains active. This particular entry is highlighted in bold in the previous table.

TABLE C.2 – SKYPE STATUS DURING INACTIVITY

NAT TABLE LEGACY STATUS								
LOCAL		OUTBOUND		REMOTE		Prot.	Tout.	Status
IP	Port	IP	Port	IP	Port			
10.10.0.101	48018	1.1.1.11	48018	217.208.205.94	39652	TCP	1800	A

After testing connection setup and keep alive, now we will proceed to establish a chat session, audio call and videoconference between *hosta* and *public* hosts. The following figures represent the result on both hosts as well as the forwarding table at that particular moment.

Private Host:

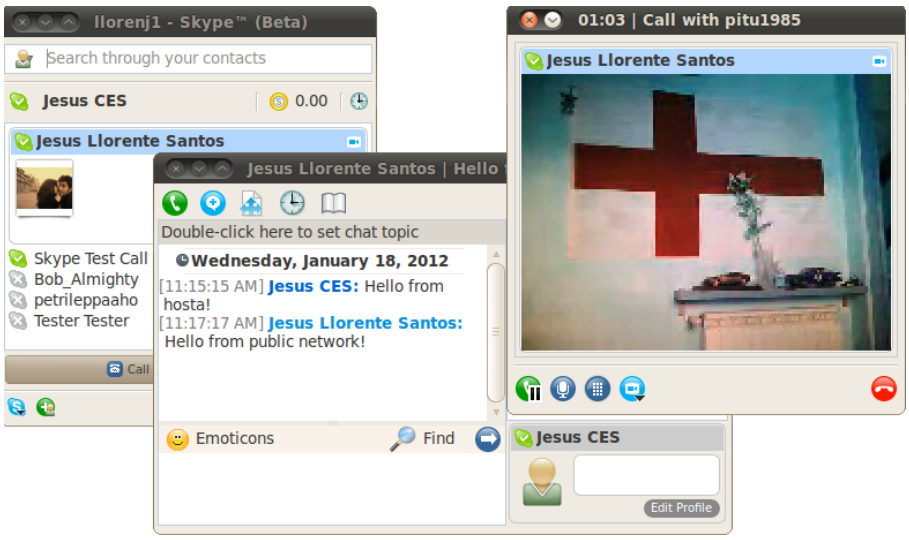


FIGURE C.1 SKYPE - HOST "HOSTA" DURING A CALL

Remote Host:

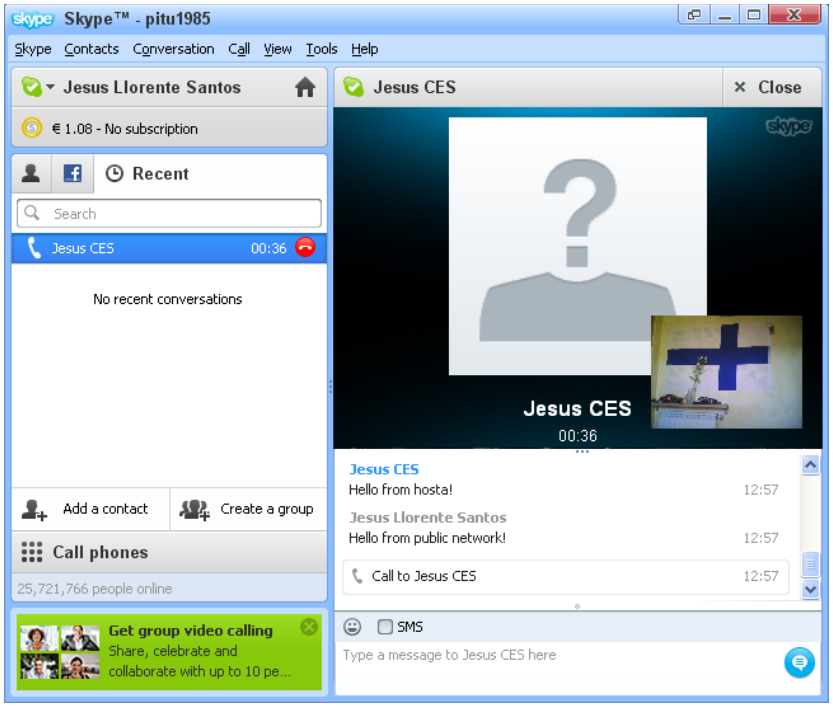


FIGURE C.2 SKYPE - HOST "PUBLIC" DURING A CALL

CES:

TABLE C.3 – SKYPE STATUS DURING A CALL

NAT TABLE LEGACY STATUS								
LOCAL		OUTBOUND		REMOTE		Prot.	Tout.	Status
IP	Port	IP	Port	IP	Port			
10.10.0.101	56813	1.1.1.11	56813	84.249.198.39	63337	TCP	1800	A
10.10.0.101	1602	1.1.1.11	1602	84.249.198.39	63337	UDP	60	A
10.10.0.101	54290	1.1.1.11	54290	83.179.24.172	22049	TCP	1800	A
10.10.0.101	1602	1.1.1.11	1602	83.179.24.172	22049	UDP	60	A
10.10.0.101	58761	1.1.1.11	58761	94.22.124.157	1925	TCP	1800	A
10.10.0.101	1602	1.1.1.11	1602	94.22.124.157	1925	UDP	60	A
10.10.0.101	58946	1.1.1.11	58946	217.209.55.157	43761	TCP	1800	A
10.10.0.101	1602	1.1.1.11	1602	217.209.55.157	43761	UDP	60	A
10.10.0.101	**48018**	**1.1.1.11**	**48018**	**217.208.205.94**	**39652**	**TCP**	**1800**	**A**
10.10.0.101	1602	1.1.1.11	1602	95.109.47.14	16989	UDP	60	A

Additional notes: Based on Figure C.1 and Figure C.2 as well as the Table C.3, we consider the test to be completely successful. Although the video functionality is only available on the remote party, Skype supports unidirectional video communication allowing *hosta* to receive the media.

Despite the figures only display the result of a single call, the actual testing carried out several audio calls, videoconferences, instant messaging and file transfers for both incoming and outgoing fashion, succeeding in all cases.

Appendix D – Application Layer Gateway for HTTP – False Start

This appendix explains how incoming HTTP and HTTPS requests are processed in order to overcome the compatibility issues that appeared during the testing process. Section 8.1.4 revealed that a host located in the public network was unable to connect properly to a web server located behind a CES.

The testing revealed that the browser was not able to download completely the whole content of the page that contained several objects. This happens because HTTP initiates multiple connections towards the same server in order to retrieve the different elements such as images or embedded objects that constitute a resource. In addition, when attempting to connect to the secured version of the pages via HTTPS within the same session it would timeout as well. The problem is that originally, CES is not able to identify these flows of information and failing to find a mapping in the forwarding table the packets are dropped. For this reason, the following Application Layer Gateway was developed for CES device.

Before describing how the process works it is worth mentioning that this application layer is triggered by an incoming TCP segment with destination port 80, 443, 8080 or 8443. These ports are well defined by IANA and are reserved for WWW services. In addition, two more databases are created and their description is the following:

Active connections: Stores information about the first connection addressed to the circular pool and subsequent connections matching the acceptance rule. The acceptance rule defines the range of remote ports that are accepted for incoming connections and a particular host. The equation is defined as follows:

$$\text{Port}_{\text{active}} - N_{\text{threshold}} \leq \text{Port}_{\text{remote}} \geq \text{Port}_{\text{active}} + N_{\text{threshold}}$$

In addition, the mapping stored for active connections corresponds to:

$$(public\_IP, public\_port, remote\_IP) \rightarrow (local\_IP, local\_port, remote\_port)$$

Allowed connections: Stores information about the rest of the supported HTTP ports that are not in use for a given remote user. The mapping stored for allowed connections corresponds to: $(public\_IP, remote\_IP) \rightarrow (local\_IP, port\_list, timeout, timestamp)$

Figure D.1 represents the scenario where a remote client attempts to connect to an HTTP service on port 80, downloads the *index.html* resource and in a parallel HTTP connection the *favicon.ico*. Then it creates a new HTTPS connection and downloads *sindex.html*. After this, an attempt to connect with a different HTTP service is received in CES and misrouted, afterwards two more connections arrive to HTTP and HTTPS services, but are dropped because they do not comply with the acceptance rule regarding port numbering.

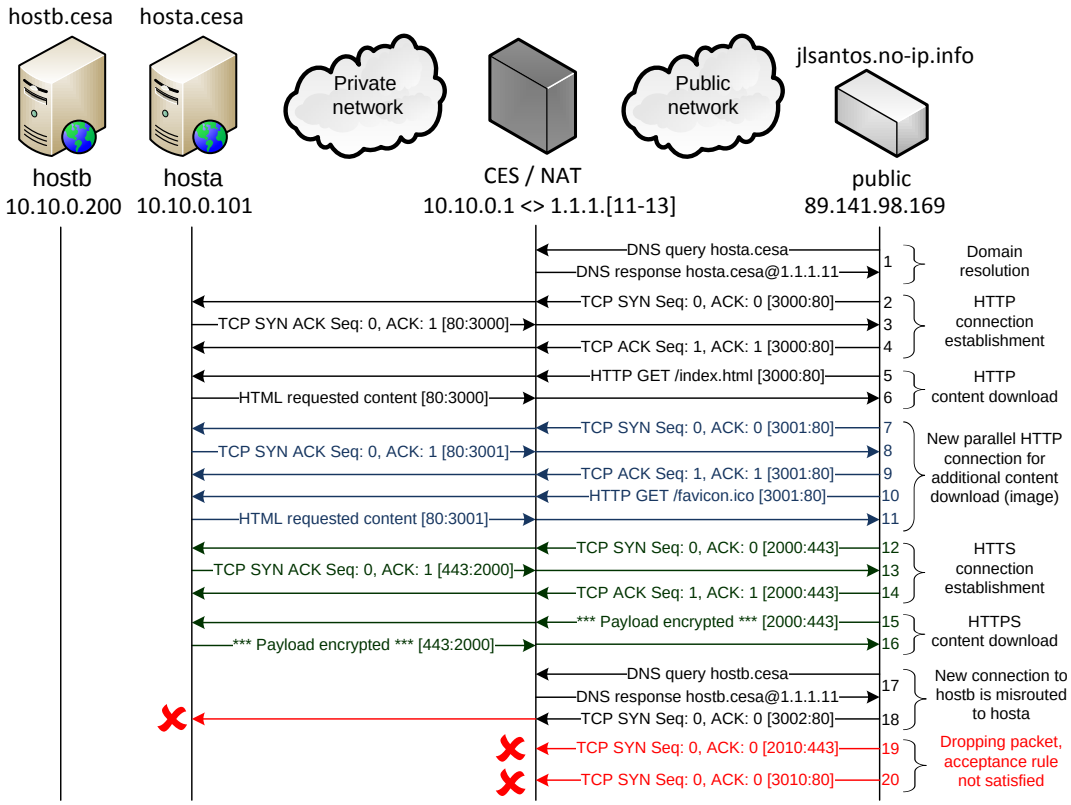


FIGURE D.1 ALG HTTP(S) - PACKET SEQUENCE

Subsequently and attending to the operation number illustrated in the previous figure, we offer a brief explanation of each one of the messages.

#1 Domain resolution request/response for the domain *hosta.cesa*. The Circular Pool creates state for the address 1.1.1.11

#2 Incoming SYN addressed to 1.1.1.11 and port 80. Application layer is triggered. There is no matching state in the forwarding table nor in active or allowed databases. The packet is returned without further action. The matching state in circular pool will forward the packet internally towards *hosta*. A new entry is added to forwarding table.

#3 Reuses the existing mapping to forward the packet towards *public*.

#4 Incoming ACK confirms the three-way handshake. There is a match in the forwarding table but it does not appear as an active connection. New entry added in active connection database as well as in allowed connections for the ports #443, 8080, 8443.

#5 #6 HTTP GET operation retrieves the content specified in the URL *index.html* with the previously established connection. There are no further modifications in the databases.

#7 There is a new HTTP connection attempt by the browser. TCP SYN is received from the remote port 3001. The packet does not match any ongoing connection in the forwarding table but satisfies a match in active connections. Moreover, the port 3001 satisfies the acceptance rule because it is within a range of ±2 from all the active ports for that communication. A new entry is added to the forwarding table and the packet is returned for further processing.

#8 Reuses the existing mapping to forward the packet towards *public*.

#9 Incoming ACK confirms the three-way handshake. There is a match in the forwarding table but it does not appear as an active connection. A new entry is added in active connection database. There is no change in the allowed connections; the ports #443, 8080, 8443 remain unchanged.

#10 #11 HTTP GET operation retrieves the content specified in the URL *favicon.ico* with the previously established connection. There are no further modifications in the databases.

#12 There is a new HTTPS connection attempt by the browser. TCP SYN is received from the remote port 2000. This packet does not match any ongoing connection in the forwarding table but a match is found in the allowed connections. The entry for allowed connections is modified and the port 443 is removed. The allowed ports are therefore #8080, 8443. A new map is added to the forwarding table and the packet is returned for further processing.

#13 Reuses the existing mapping to forward the packet towards *public*.

#14 Incoming ACK confirms the three-way handshake. There is a match in the forwarding table but it does not appear as an active connection. A new entry is added in the active connection database. There is no change in the allowed connections; the ports #8080, 8443 remain unchanged.

#15 #16 HTTPS GET operation retrieves the content specified in the URL *sindex.html* with the previously established connection. There are no further modifications in the databases.

#17 Domain resolution request/response for the domain *hosta.cesa*. The Circular Pool creates state for the address 1.1.1.11 that matches the same address in use for previous HTTP connection to *hosta.cesa*.

#18 There is a new HTTP connection attempt by the browser. TCP SYN is received from the remote port 3002. This packet does not match any ongoing connection in the forwarding table but a match is found in the active connections. Moreover, the port 3002 satisfies the acceptance rule because it is within a range of ±2 from all the active ports for that communication. A new map is added to the forwarding table and the packet is returned for further processing to be delivered to *hosta.cesa*. This mapping is incorrect because the intended recipient is indeed *hostb.cesa*. As a consequence the packet is misrouted to *hosta* instead of *hostb*.

#19 There is a new HTTPS connection attempt by the browser. TCP SYN is received from the remote port 2010. This packet does not match any ongoing connection in the forwarding table but satisfies a match in the active connections. Failing to satisfy the acceptance rule because port 2010 is out of the ±2 threshold from 2001 (best case scenario) the packet is dropped.

#20 Likewise case #19, this HTTP connection fails to satisfy the acceptance rule for the port 3010 based on the ±2 threshold from 3002 and accordingly the packet is dropped.

Figure D.2 represents the flow diagram of the HTTP(S) application layer.

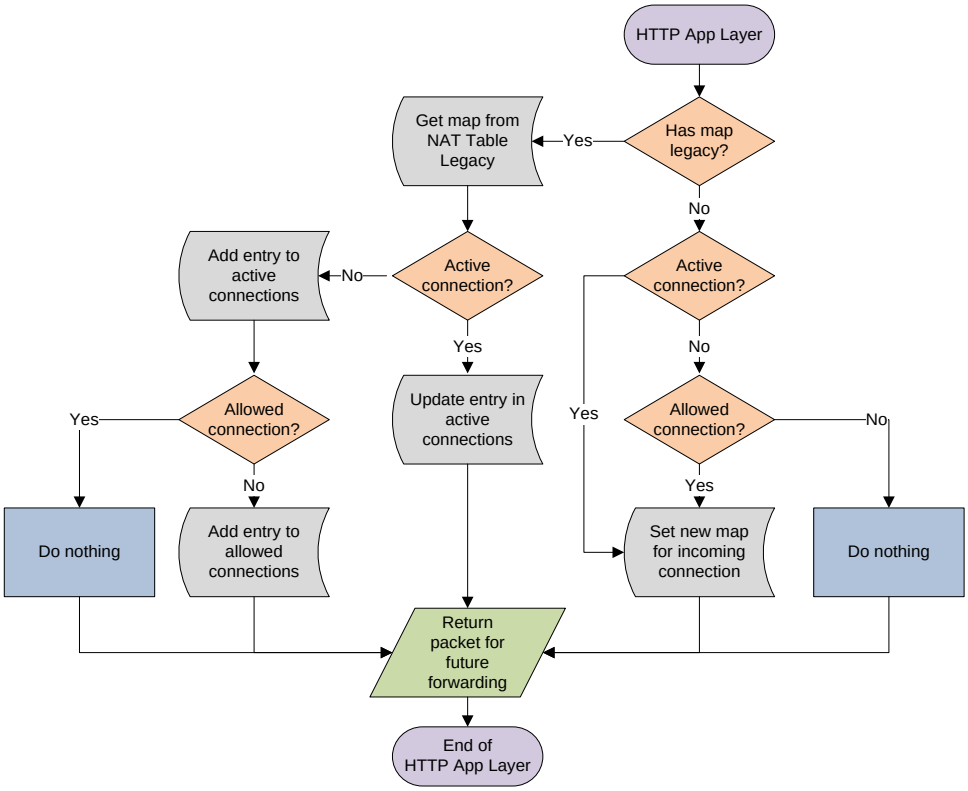


FIGURE D.2 ALG HTTP(S) - FLOW DIAGRAM

An example to confirm the behavior of the application layer is introduced here. During this test we will connect from *public* to the URL http://hosta.cesa:8080/ces.html via HTTP and once the page has loaded will attempt to connect to https://hosta.cesa:8443/ces_secure.html via HTTPS within the same session. Below we present the result of the operation on *public* host in Figure D.3 and Figure D.4. Also the forwarding information in CES is represented in Table D.1.

Connecting to HTTP Service

FIGURE D.3 ALG HTTP - WEB BROWSER – HOST "PUBLIC" AND HTTP TO "HOSTA"

Connecting to HTTPS Service

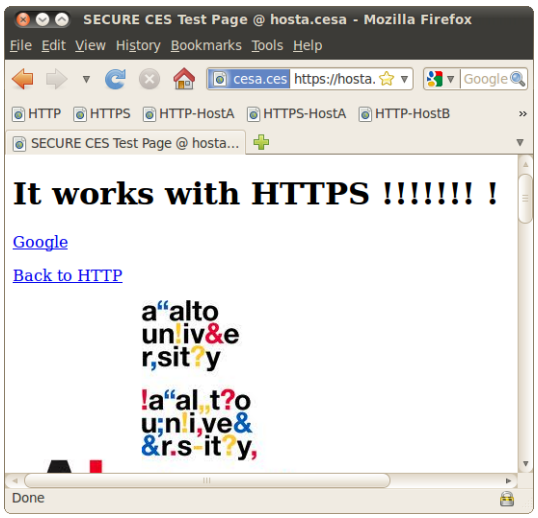


FIGURE D.4 ALG HTTP - WEB BROWSER – HOST "PUBLIC" AND HTTPS TO "HOSTA"

CES:

TABLE D.1 – HTTP & HTTPS INCOMING CONNECTIONS WITH ALG HTTP(S)

NAT TABLE LEGACY STATUS								
LOCAL		OUTBOUND		REMOTE		Prot.	Tout.	Status
IP	Port	IP	Port	IP	Port			
10.10.0.101	8080	1.1.1.11	8080	89.141.98.169	42378	TCP	12	A
10.10.0.101	8080	1.1.1.11	8080	89.141.98.169	42379	TCP	12	A
10.10.0.101	8080	1.1.1.11	8080	89.141.98.169	42380	TCP	12	A
10.10.0.101	8080	1.1.1.11	8080	89.141.98.169	42381	TCP	12	A
10.10.0.101	8080	1.1.1.11	8080	89.141.98.169	42382	TCP	12	A
10.10.0.101	8443	1.1.1.13	8443	89.141.98.169	58706	TCP	1800	A
10.10.0.101	8443	1.1.1.13	8443	89.141.98.169	58707	TCP	1800	A
10.10.0.101	8443	1.1.1.13	8443	89.141.98.169	58708	TCP	1800	A
10.10.0.101	8443	1.1.1.13	8443	89.141.98.169	58709	TCP	1800	A
10.10.0.101	8443	1.1.1.13	8443	89.141.98.169	58710	TCP	1800	A

Additional notes: As we can observe based on the output produced by the *public* host both HTTP and HTTPS operations were successful. The new mechanisms introduced by the application layer enable CES to identify the active connections. By establishing a rather limited threshold we can predict with high certainty future incoming connections and create a mapping on the fly in order to forward those data packets to the private host.

Caveats: The testing environments as well as the application layer have been designed to accommodate and serve the HTTP(S) connections originating only from a single host on the public domain. In spite of this, there is still a possibility that under certain circumstances the application layer fails to operate successfully. The problems found can be classified under two categories, inherent to NAT(ed) connections on the originating side and misrouting in the private network.

NAT problem: The application layer implements a basic heuristic process based on the TCP source port to determine if a connection comes from a known user to perform packet delivery. The issue here is that NATs can modify this value according to its local forwarding table. As a consequence, the connection could be dropped.

Packet misrouting: Another issue with the heuristic consequently sets a wrong mapping in the forwarding table in such a way that a packet intended for hostb ends up being forwarded to hosta. This happens because the new connection originating in public "overlaps" with previous ongoing connection from the same host towards hosta. The term overlap indicates that the new connection is understood and processed by the CES as thought it belonged to the ongoing ones.

For these reasons it seems natural to assume that the application layer works under certain network conditions but *fails* to operate successfully under many others, preventing users from establishing connections with their intended service/device. The heuristic methods implemented cannot contribute enough to adapt to all scenarios.